MOBILEMAN

IST-2001-38113

Mobile Metropolitan Ad hoc Networks

MOBILEMAN
Architecture, protocols and services

Deliverable D13

Contractual Report Preparation Date: March 2005
Actual Date of Delivery: 10 June 2005 (*)
Estimated Person Months: 10
Number of pages: 245

Contributing Partners: Consiglio Nazionale delle Ricerche (Italy), University of Cambridge (UK), Institut Eurecom (France), Helsinki University (Finland), NETikos (Italy), Scuola Universitaria Professionale della Svizzera Italiana (Switzerland)

Authors: Eleonora Borgia, Marco Conti, Franca Delmastro, Giovanni Mainetto, Gaia Maselli, Giovanni Turi (CNR), Jon Crowcroft, Andrea Passarella (Cambridge), Claudio Lavecchia, Pietro Michiardi, Refik Molva (Eurecom), Jose Costa Requena (HUT), Cremonese, Veronica Vanni (Netikos) Ralph Bernasconi, Claudia Brazzola, Ivan Defilippis, Jennifer Duyne, Silvia Giordano, Alessandro Puiatti, (SUPSI)

(*) new schedule accepted by the Project Officer

Abstract: The aim of this deliverable is to present the consolidated architecture, protocols and services designed for the MobileMAN project. First, we present the complete architecture with the communication flows among different functions then we discuss protocols belonging to the MobileMAN protocols' stack. Protocols are presented by following a bottom up approach from wireless technologies up to the application and economic issues.

Project funded by the European Community under the "Information Society Technologies" Programme (1998-2002)

Summary

This document presents the consolidated MobileMAN architecture and protocols (from the medium access control to the application layer). The presentation follows a bottom up approach from wireless technologies up to application and economic issues. When appropriate the social and economic perspectives are also used to compare and contrast technical solutions.

Deliverable D13 is the third document in a series (D5, D10 and D13) devoted to present the MobileMAN architecture and protocols. To make reading easier, we decided to include in D13 all the material relevant to understanding the MobileMAN architecture. Thus, part of the material presented in D5 and D10 is replicated here. This makes D13 a self-contained presentation of MobileMAN architecture and protocols. Hereafter, we present the organization of the deliverable and explain the new contributions and results that are the outcome of the last 8 months of project activities (and therefore, not included in the previous deliverables).

The MobileMAN architecture supporting cross-layer interactions is presented in Section 1. In this section we focus on the specification of the Network Status (NeSt) which is a node local memory where information gathered at different layers of the network stack is shared among different protocols and used to adapt the behavior of the node depending on the particular circumstance (e.g., traffic type, channel perturbations, network status, node selfishness and/or maliciousness, among the others) the node operates in. With respect to D10, a new section (Section 1.5) has been added to D13 to provide a description of our software architecture implementing the NeSt. Specifically, in Section 1.5, we present the software architecture of a NeSt prototype supporting cross-layer interactions between a proactive routing protocol and the middleware platform, CrossROAD, which has been developed during the project.

Section 2 discusses the problems when using 802.11 cards in multi-hop ad hoc networks and the enhanced card we designed and implemented to solve these problems. No new contributions are presented in this section. The description of the enhanced card has been presented in Deliverable D12.

Section 3 is devoted to presenting MobileMAN networking protocols that use the one-hop transmission services provided by the network interface card to construct end-to-end (reliable) delivery services. The basic functionalities implemented by these protocols include routing and forwarding algorithms to deliver the information through the MANET. In addition the low reliability of communications (due to wireless communications, users' mobility, etc.), and the possibility of network congestion require a transport protocol tuned for

the MANET environment. Last, to efficiently support cross-layer interactions the routing algorithms have to be extended to support a *location service* to discover the nodes in the network that are offering a specified service. All these functionalities and protocols were already specified in D5 and D10. The new material presented in this section is related to the validation of the transport protocol mechanisms we developed (see the TPA protocol in Deliverable D5). This validation, performed via simulation, shows that TPA outperforms legacy TCP protocol in all operating conditions we investigated.

Section 4 addresses the enforcement of cooperation within a MANET. The deliverable proposes an approach that analyzes the implications of the lack of cooperation in a peer-to-peer network together with a sociological study of cooperation models. Sections 4.6 and 4.7 present the new contributions. Section 4.6 completes the theoretical study of cooperation models developed in deliverable D10, while Section 4.7 contains the application of such concepts to existing peer-to-peer groups - that is, users of file-sharing applications over the internet. Analogies between these groups and a MobileMAN users group are underlined and some conclusions that can apply to the case of MobileMAN are drawn.

Section 5 deals with the MobileMAN middleware platforms. Firstly, we present the Pastry platform that was identified in Deliverable D5 as the most interesting middleware platform (among existing p2p platforms) for a MobileMAN network. Then, we introduce and describe *CrossROAD*: *CROSS-layer Ring Overlay for AD hoc networks.* CrossROAD is our proposal to enhance Pastry by exploiting cross-layer interactions. The new material we present in D13 is the detailed description of the CrossROAD software architecture.

In Section 6 we present the three applications we selected to test the MobileMAN architecture: UDDI, a whiteboard application (WB) and a VoIP session. The three applications were already presented in Deliverable D10. The new material presented in D13 is related to UDDI and WB. Both applications were integrated in the MobileMAN architecture by exploiting the services offered by Pastry (see D10). During the third year of the project we investigated the integration of UDDI and WB with CrossROAD in order to exploit the better quality of service possible by cross-layer optimizations.

No new material as been added to Section 7 and Section 8. Specifically, Section 7 investigates business models for the MobileMAN project. In Section 8 we highlight the potentialities of the ad hoc paradigm in opportunistic networking scenarios.

Table of Contents

1. MOBILEMAN ARCHITECTURE

The cross-layer approach highlighted in the Deliverable D5 to optimize the functioning of the system architecture (by exploiting cross-layering interactions still maintaining full compliance with the layer separation principle) was recognized by all partners, and by the project reviewers, as interesting and worth exploiting. However, as pointed out in the Deliverable D4, we are pioneers in this area, and to reduce the risks, we decided i) to continue, as planned in Annex 1, to investigate the ad hoc networking paradigm following a layered legacy architecture and, in parallel, ii) to explore (as much as possible depending on the available resources) the potentialities of a cross-layer architecture.
In this section we present the MobileMAN architecture. It is based on a layered approach, which can be enhanced with cross layering interactions if information gathered at different layers of the network stack is shared in a common local memory structure (Network Status, referred to as *NeSt*).

In the following sections we first present the rationale behind our *loosely-coupled* cross layering approach, and then we present the resulting MobileMAN architecture. The second part of the section focuses on the main element of the cross layer architecture: the NeSt. Specifically we introduce the definition of the NeSt interaction models, and define the exported interface.

1.1. Toward loosely-coupled cross-layering

The Internet transparently connects millions of heterogeneous devices, supporting a huge variety of communications. From a networking standpoint, its popularity is due to a core design that has made it extensible, and robust against evolving usage, as well as failures. Now, mobile devices and wireless communications prompt the vision of networking without a network (ad hoc networking). This brings new challenging issues where the need for flexibility confronts ad hoc constraints. A careful architectural design for the ad hoc protocol stack is necessary to incorporate this emerging technology.

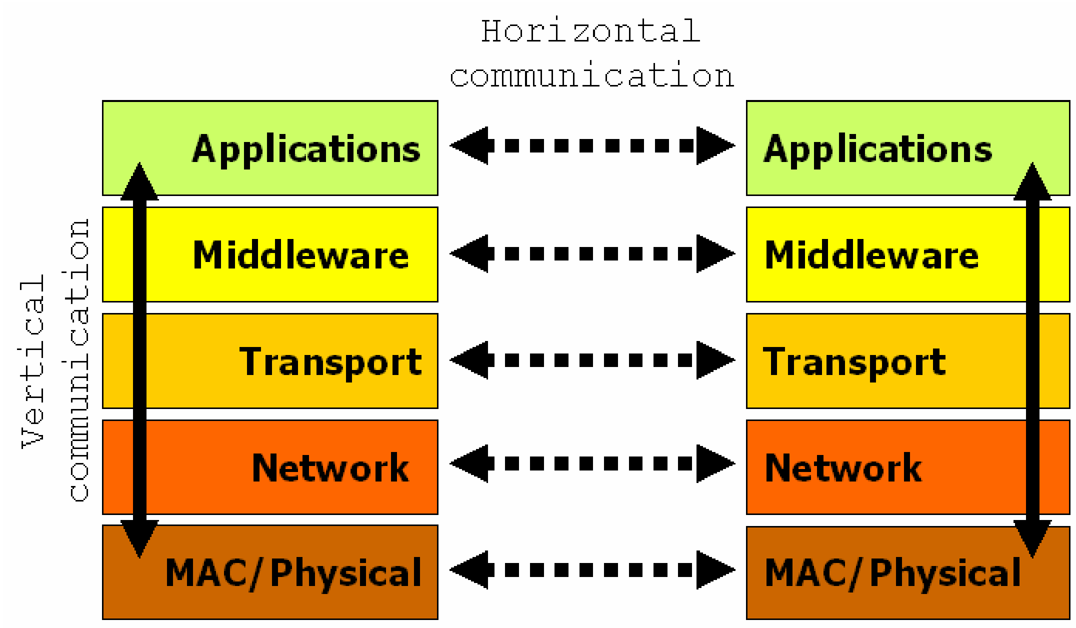


Figure 1.1. The Internet emphasizes horizontal communication between peer protocol layers to save routers resources, while ad hoc networking promotes vertical interaction to conserve bandwidth.

The Internet architecture layers protocol and network responsibilities, breaking down the networking system into modular components, and allowing for transparent improvements of single modules. In a *strict-layered* system, protocols are independent of each other and interact through well-defined (and static) interfaces: each layer implementation depends on the interfaces available from the lower layer, and those exported to the upper layer. Strict-layering provides flexibility to a system's architecture: extensions introduced into single levels do not affect the rest of the system. The separation of concerns brings the added benefits of minimizing development costs by re-using existing code. This design approach relies on "horizontal" communication between peer protocol layers on the sender and receiver devices (the dashed arrows in Figure 1.1). The result is a trend to spend bandwidth (an abundant resource in the Internet) instead of processing power and storage.

Several aspects of the Internet architecture have led to the adoption of this strict-layer approach also for mobile ad hoc networks. Some of these aspects are: i) the "IP-centric" view of ad hoc networks; ii) the flexibility offered by independent layers, which allows for reuse of existing software. The choice of the layered approach is re-enforced by the fact that ad hoc networks are considered mobile extensions of the Internet, and hence the protocol stack has to be suitable. However, this design principle clashes with the following facts: i) issues like energy management, security, and cooperation characterize the whole stack, and cannot be solved inside a single layer; ii) ad hoc networks and the Internet have conflicting constraints, and while the former are dynamic, the latter is relatively static.

Some guidelines to approach these problems point to an enhancement of "vertical" communication in a protocol stack [1],[2] (see Figure 1.1), as a way to reduce peer (horizontal) communication, and hence conserve bandwidth. Vertical communication, especially between non-adjacent layers, facilitates local data retrieval, otherwise carried out through network communication. The practice of accessing not only the next lower layer, but also other layers of the protocol stack leads to *cross-layering*, to allow performance improvements. The main downside of strict-layering is that it hinders extensibility: a new higher-level component can only build on what is provided by the next lower layer [3]. Hence, if one layer needs to access functionality or information provided by a non-adjacent layer, then an intermediate extension needs to be devised. Cross-layering allows non-adjacent protocols to directly interact, making overall optimizations possible, and achieving extensibility at the eventual expense of flexibility.

In the literature there is much work showing the potential of cross-layering for isolated performance improvements in ad hoc networks. However, the focus of that work is on specific problems, as it looks at the joint design of two-to-three layers only. For example, in [4] cross-layer interactions between the routing and the middleware layers allow the two levels to share information with each other through system profiles, in order to achieve high quality in accessing data. An analogous example is given in [5], where a direct interaction between the network and the middleware layers, termed Mobile Peer Control Protocol, is used to push a reactive routing to maintain existing routes to members of a peer-to-peer overlay network. In [6] the authors propose an interaction between the MAC and routing layers, where information like Signal-to-Noise Ratio, link capacity and MAC packet delay is communicated to the routing protocol for the selection of optimal routes. Another example is the joint balancing of optimal congestion control at the transport layer with power control at the physical layer proposed in [7]. This work observes how congestion control is solved in the Internet at the transport layer, assuming link capacities to be fixed quantities. In ad hoc networks, this is not a good assumption, as transmission power, and hence throughput, can be dynamically adapted on each link. Last, but not least, [8] proposes cross-layer interaction between physical, MAC and routing layers, to perform joint power control and link scheduling as an optimized objective.

Although these solutions are clear examples of optimization introduced by cross-layering, the drawback on the resulting systems is that they contain tightly coupled and therefore mutually dependent components. Additionally, while an individual suggestion for cross-layer design, in isolation, may appear appealing, combining them all together could result in interference among the various optimizations [9]. From an architectural point of view this approach leads to an "unbridled" stack design, hard to maintain efficiently, because every modification must be propagated across all protocols. To give an example of interfering optimizations, let us consider an adaptation loop between a rate adaptive MAC and minimal hop routing protocol (most ad hoc routing protocols are minimum hop). A rate adaptive MAC would be able to analyze the quality of channels, suggesting higher layers on the outgoing links, which provide the higher data rates in correspondence with shorter distances. This conflicts with typical decisions of a minimum hop routing protocol, which chooses longer link (for which the signal strength and data rate are typically lower) to reach the destination while using as few hops as possible.

We claim that cross-layering can be achieved maintaining the layer separation principle, with the introduction of a vertical module, called *Network Status*[1] (NeSt), which controls all cross-layer interactions (see Deliverable D5). The NeSt aims at generalizing and abstracting vertical communications, getting rid of the tight-coupling from an architectural standpoint. The key aspect is that protocols are still implemented in isolation inside each layer, offering the advantages of:

- allowing for full compatibility with standards, as NeSt does not modify each layer's core functions;
- providing a robust upgrade environment, which allows the addition or removal of protocols belonging to different layers from the stack, without modifying operations at other layers;
- maintaining the benefits of a modular architecture (layer separation is achieved by standardizing access to the NeSt).

Besides the advantages of a full cross-layer design, which still satisfies the layer-separation principle, the NeSt provides full context awareness at all layers. Information regarding the network topology, energy level, local position, etc., is made available by the NeSt to all layers, in order to achieve optimizations, and offer performance gains from an overhead point of view. Although this awareness is restricted to the node's local view, protocols can be designed so as to adapt the system to highly variable network conditions (the typical ad hoc characteristic).

This innovative architecture opens research opportunities for techniques to design and evaluate new ad hoc protocols (see Section 1.3.1), but also remains compliant with the usage of legacy implementations, introducing new challenging issues concerning the usage of cross-layering for the Internet more generally (see Section 1.4).

Relaxing the Internet layered architecture, by removing strict layer boundaries, is therefore an open issue in the mobile ad hoc networks evolution. However, the layered approach was, and is, one of the key elements of the world-wide diffusion of the Internet. The question is to what extent the pure layered approach needs to be modified.

One of the main problems caused by direct cross-layer interactions (as already discussed in Deliverable D5) is the resulting *tight-coupling* of interested entities. To solve this problem, the NeSt stands vertically beside the network stack (as shown in Figure 1.2) handling eventual cross-layer interactions among protocols. In other words, the NeSt plays the role of intermediary, providing standard models to design protocol interactions. While the new component uniformly manages vertical exchange of information between protocols, usual network functions still take place layer-by-layer through standardized interfaces, which remain unaltered. This introduced level of indirection maintains the *loosely-coupling* characteristic of Internet protocols, preserving the flexible nature of a layered architecture.

[1] This name indicates the collection of network information which a node gathers at all layers. It should not be confused with a concept of globally shared network context.

The idea is to have the NeSt exporting an interface toward protocols, so as to allow sharing of information and reaction to particular events. In this way, cross-layer interactions do not directly take place between the interested protocols, but are implemented using the abstractions exported by the NeSt. This approach allows protocols' designers to handle new cross-layer interactions apart, without modifying the interfaces between adjacent layers.

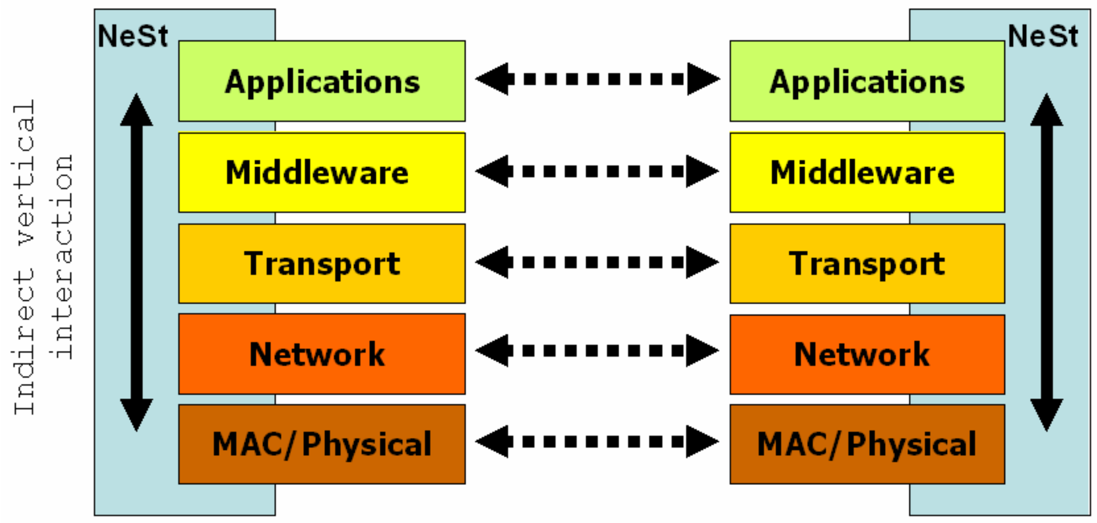


Figure 1.2. An architectural trade-off for loosely-coupled vertical protocol interactions.

1.2. MobileMAN Architecture

The resulting MobileMAN architecture is shown in Figure 1.3 below. Specifically, we have a basic architecture that follows the original reference architecture defined in Annex 1. This architecture can be enhanced with cross-layer interactions if the *NeSt* is implemented and protocols implement the interfaces to interact with it.

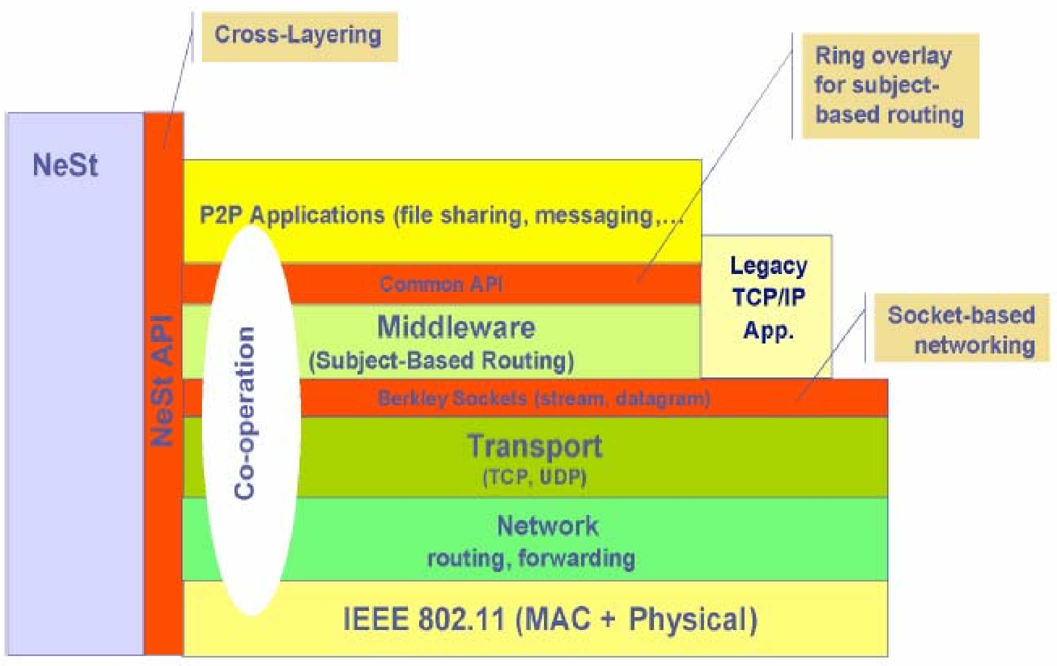

Cross-Layering
Ring overlay for subject-based routing
NeSt
NeSt API
P2P Applications (file sharing, messaging,...)
Common API
Middleware (Subject-Based Routing)
Legacy TCP/IP App.
Socket-based networking
Co-operation
Berkley Sockets (stream, datagram)
Transport (TCP, UDP)
Network routing, forwarding
IEEE 802.11 (MAC + Physical)

Figure 1.3: MobileMAN Architecture

In this deliverable the components of the above architecture are analyzed in detail. In the next subsection we provide a detailed presentation of the NeSt, while

- The MAC and physical layers, based on 802.11 technology, are analyzed in Section 2.
- Routing related functionalities are discussed in Section 3, while in Section 4 the cooperation mechanism and models are addressed.
- Middleware solutions are analyzed in Section 5.
- Applications and usage scenarios are discussed in Sections 6 and 7, respectively.

1.3. NeSt functionalities

The *Network Status*[2] (NeSt) is the basic element we have designed to introduce cross-layering interactions in the MobileMAN architecture still maintaining the layer separation principle. To this end the NeSt stands vertically beside the network stack handling eventual cross-layer interactions among protocols. The idea is to have the NeSt exporting an interface toward protocols, so as to allow sharing of information and reaction to particular events. The work described in [10] [11] and Deliverable D5 introduces this idea. The work performed during the second year completes the NeSt specification by defining interaction models, presenting the exported interface [24].

The NeSt supports cross-layering implementing with two models of interaction between protocols: *synchronous* and *asynchronous*. Protocols interact synchronously when they share private data (i.e. internal status collected during their normal functioning). A request for private data takes place on-demand, with a protocol querying the NeSt to retrieve data produced at other layers, and waiting for the result. Asynchronous interactions characterize the occurrence of specified conditions, to which protocols may be willing to react. As such conditions are occasional (i.e. not deliberate), protocols are required to subscribe for their occurrences. In other words, protocols subscribe for events they are interested in, and then return to their work. The NeSt in turn is responsible for delivering eventual occurrences to the right subscribers. Specifically, we consider two types of events: *internal* and *external*. Internal events are directly generated inside the protocols. Picking just one example, the routing protocol notifies the rest of the stack about a ``broken route" event, whenever it discovers the failure of a preexisting route. On the other side, external events are discovered inside the NeSt on the basis of instructions provided by subscriber protocols. An example of external event is a condition on the host energy level. A protocol can subscribe for a ``battery-low" event, specifying an energy threshold to the NeSt, which in turn will notify the protocol when the battery power falls below the given value.

As the NeSt represents a level of indirection in the treatment of cross-layer interactions, an agreement for common-data and events representation inside the vertical component is a fundamental requirement. Protocols have to agree about a common representation of shared information, in order to guarantee loosely-coupling. To this end, the NeSt works with *abstractions* of data and events, intended as a set of data structures that comprehensively reflect the relevant (from a cross-layering standpoint) information and special conditions used throughout the stack. A straightforward example is the topology information collected by a routing protocol. In order to abstract from implementation details of particular routing protocols, topology data can be represented as a graph inside the NeSt. Therefore, the NeSt becomes the provider of shared data, which appear independent of its origin and hence usable by each protocol.

How is protocols internal data exported into NeSt abstractions? The NeSt accomplishes this task by using *call-back* functions, which are defined and installed by protocols themselves. A call-back is a procedure that is registered to a library (the NeSt interface) at one point in time, and later on invoked (by the NeSt). Each call-back contains the instructions to encode private

[2] This name indicates the collection of network information which a node gathers at all layers. It should not be confused with a concept of globally shared network context.

data into an associated NeSt abstraction. In this way, the protocol designer provides a tool for transparently accessing protocol internal data.

1.3.1. The NeSt interface

In order to give a technical view of the vertical functionalities, we assume that the language used by the NeSt to interface the protocol stack allows for declaration of functions, procedures and common data structures. We adopt the following notation to describe the NeSt interface:

functionName: *(input)* → *output*

Each protocol starts its interaction with the NeSt by *registering* to the vertical component. This operation assigns to each protocol a unique identifiers (PID), as shown by step *a* in Figure 1.4. NeSt functionalities: register and seize. The registration is expected to happen once for all at protocol bootstrap time by calling

register: *()* → *PID*

As described in the previous Section, the NeSt does not generate shared data, but acts as intermediary between protocols. More precisely, a protocol *seizes* the NeSt abstractions related to its internal functionalities and data structures. The example of the network topology suggests the routing protocol to acquire ownership of an abstract graph containing the collected routing information. This operation requires a protocol to identify itself, providing the PID, and to specify the abstraction's identifier (AID) together with the associated call-back function (see step *b* in Figure 1.4). When invoked, the call-back function fills out the abstraction, encoding protocol internal representation in NeSt format. Please note that the call-back invocation takes place asynchronously with the seizing operation, every time a fresh copy of the associated data is needed inside the NeSt. The whole process begins by calling

seize: *(PID, AID, readCallBack())* → *result*

The result of a call to *seize()* indicates the outcome of the ownership request.

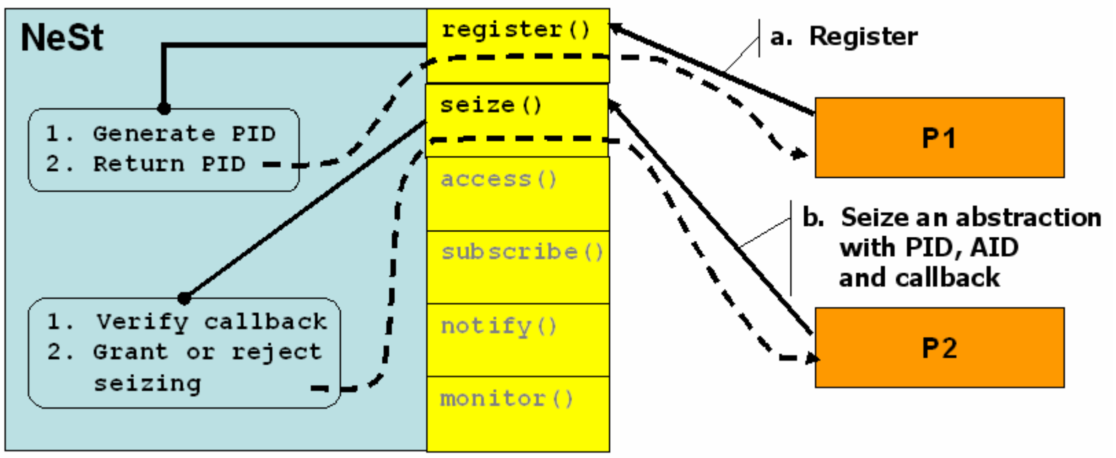


Figure 1.4. NeSt functionalities: register and seize.

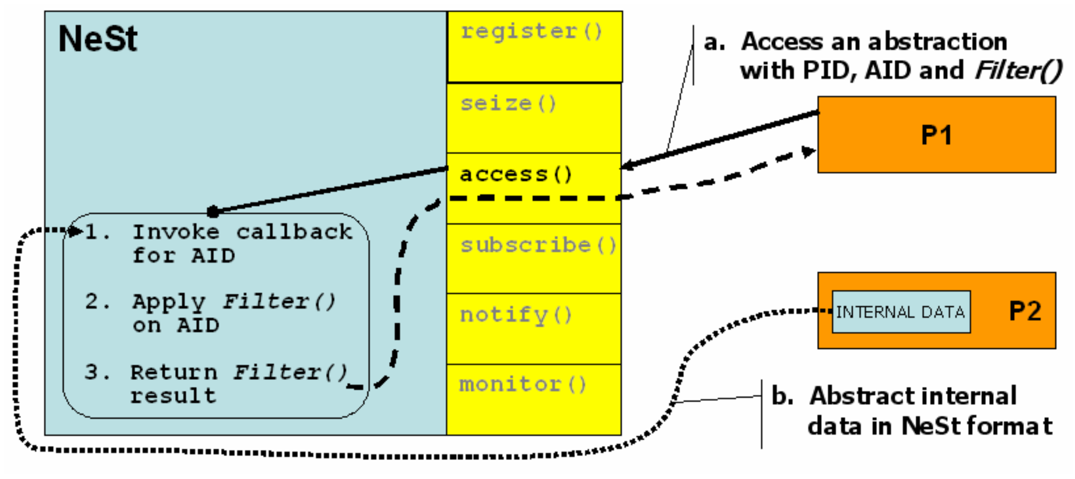

NeSt
register()
seize()
access()
subscribe()
notify()
monitor()
1. Invoke callback for AID
2. Apply Filter() on AID
3. Return Filter() result
a. Access an abstraction with PID, AID and Filter()
P1
INTERNAL DATA
P2
b. Abstract internal data in NeSt format

Figure 1.5. NeSt functionalities: access an abstraction.

Once an abstraction has been seized, the NeSt is able to satisfy queries of interested entities. A protocol *accesses* an abstraction by calling

access: *(PID, AID, filter()) → result*

This function shows that the caller has to identify itself with a valid PID, providing also the abstraction identifier and a *filter* function. The latter parameter is a container of instructions for analyzing and selecting only information relevant to the caller's needs. The NeSt executes this call by spawning an internal computation that performs the following steps (see Figure 1.5):

Invoke the call-back installed by the abstraction's owner (if any).

Filter the returned data locally (i.e. in the context of the NeSt).

Deliver the filtering result to the caller.

The remaining functions of the NeSt interface cope with asynchronous interactions.

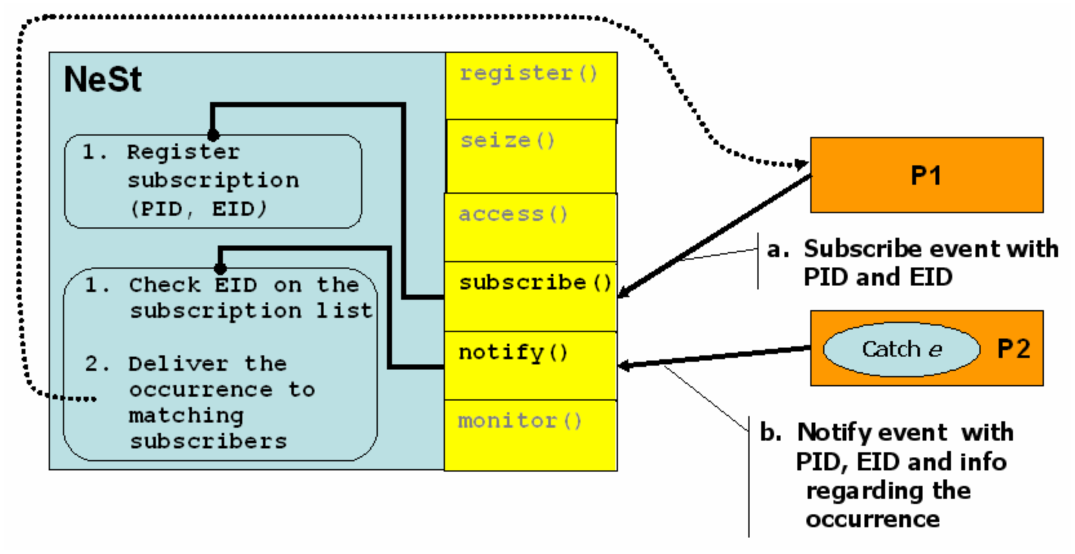

NeSt
register()
seize()
access()
subscribe()
notify()
monitor()
1. Register subscription (PID, EID)
1. Check EID on the subscription list
2. Deliver the occurrence to matching subscribers
P1
a. Subscribe event with PID and EID
Catch e
P2
b. Notify event with PID, EID and info regarding the occurrence

Figure 1.6. NeSt functionalities: management of internal events.

In the case of internal events, the role of the NeSt is to collect subscriptions, wait for notifications, and vertically dispatch occurrences to the appropriate subscribers, as shown in Figure 1.6. A protocol *subscribes* for an event by identifying itself and providing the event's identifier (EID), calling the function

subscribe: *(PID, EID) → result*

To *notify* the occurrence of an event, a protocol has to specify in addition to the event identifier (EID), information regarding the occurrence. This happens by calling the function

notify: *(PID, EID, info)* → *result*

After the notification of an event, the NeSt checks it against subscriptions, and dispatches the occurrence to each subscriber.

In the case of external events, protocols subscribe by instructing the NeSt on how to detect the event. The rules to detect an external event are represented by a monitor function that periodically checks the status of a NeSt abstraction. When the monitor detects the specified condition, the NeSt dispatches the information to the subscriber protocol.

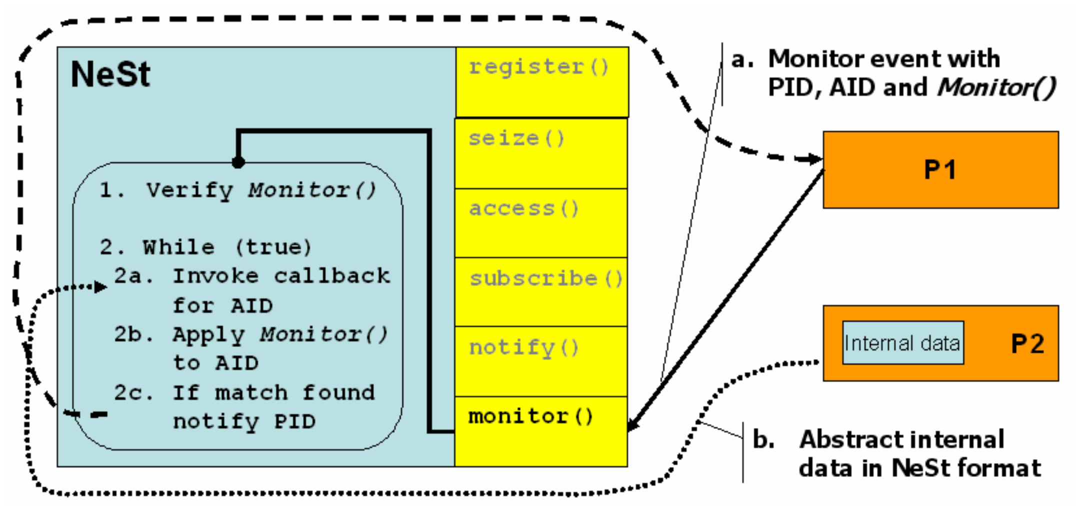

NeSt
register()
seize()
access()
subscribe()
notify()
monitor()
1. Verify Monitor()
2. While (true)
2a. Invoke callback for AID
2b. Apply Monitor() to AID
2c. If match found notify PID
a. Monitor event with PID, AID and Monitor()
P1
Internal data
P2
b. Abstract internal data in NeSt format

Figure 1.7. NeSt functionalities: management of external events.

As shown in Figure 1.7. NeSt functionalities: management of external events., a protocol delegates the *monitoring* of an external event by passing to the NeSt a monitor function and the identifier of the target abstraction. This happens by calling

monitor: *(PID, AID, monitor())* → *result*

The NeSt serves this call by spawning a *persistent* computation (see Figure 1.7) that executes the following steps:

Verify the monitor (e.g. type checking);

While (true)

Refresh the abstraction invoking the associated call-back.

Apply the monitor to the resulting content.

If the monitor detects the special condition, then notify the requesting protocol.

The result of a call to *monitor()* only returns the outcome of the monitor's installation, while the notification of external events takes place asynchronously.

1.3.2. Design and implementation remarks

It is difficult to find comparison to the proposed architecture as, to the best of our knowledge, there are no similar approaches in the organization of a protocol stack. Yet, there are important observations and remarks to be given.

First of all, the NeSt is a component dedicated to enabling optimization. If on the one hand it helps maintaining the layering principle allowing loosely-coupled interactions, on the other one it has to guarantee the appropriate level of real-timing. In other words, when subjected to a heavy load of cross-layering, the NeSt should be responsive, avoiding making protocols efforts fruitless. For example, in the case of synchronous interactions where call-backs are employed, the NeSt should not degrade the performance of both requestor and provider protocols. For these reasons, it advisable to pre-fetch and cache exported data (when possible), serving a series of accesses to the same abstraction with fewer call-backs executions. However, this approach also requires the presence of cache invalidation mechanisms, which protocols can use to stale pre-fetched or cached abstractions.

As presented here, the NeSt should come with an *a priori* set of abstractions for data and events, to which protocols adapt in order to cross-interact. A more mature and desirable approach, would reverse the adaptation process, having the vertical component adapting to whatever protocols provide. For example, this adaptation issue could be solved through the usage of *reflection*, a characteristic of some modern programming languages [12] that enables introspection of software components, allowing for dynamic changes in behavior. A NeSt reflective API would allow each protocol to define its contribution to cross-layer interactions, providing an initial registration of profiles describing the data and the events it is able to share. The resulting data and event sharing would be more *content-based* than the presented *subject-based* mechanism. With this approach, the sole agreement between the two parties would regard the representation of protocol profiles. A solution could be the usage of a language which provides rules to define both profiles data and meta-data, like for example the eXtesible Mark-up Language (XML). This solution would restrict the agreement on the set of tags (i.e. the *grammar*) to be used in building profiles. Please note that such a usage of higher level programming languages would interest only initial negotiation phases between the NeSt and the protocols, without affecting the run-time performance.

One may argue that the NeSt exhibits some conceptual similarities with a Management Information Base (MIB). A MIB is a collection of network-management information that can be accessed, for example, through the Simple Network Management Protocol (SNMP). SNMP facilitates the exchange of information between network devices and enables network administrators to manage performances, find and solve problems, and plan for network growth. Some NeSt functionalities could be realized through a local MIB (storing protocols information), to which other protocols can access in order to read and write data. However, the NeSt and MIBs are targeted to different goals. MIBs are designed for network statistics and *remote* management purposes, while the NeSt aims at overall *local* performance improvements. Furthermore, the MIB's nature makes it not suitable for the real-time tasks typical of NeSt optimizations, which require only local accesses and fine-grained time scales (e.g. in the order of single packets sent/received).

1.4. Examples of cross-layer interactions

The NeSt architecture, as described in the previous Sections, is a *full* cross-layer approach where protocols become adaptive to both application and underlying network conditions. Such an approach brings the stack as a whole to the best operating trade-off. This has been highlighted in [13], where the authors point at global system requirements, like energy saving and mobility management, as design guidelines for a joint optimization. Our approach opens different perspectives in the evaluation of network protocols. We claim that in a full cross-layer framework, the performance of a protocol should not only be evaluated by looking at its particular functionalities, but also by studying its contribution in cross-layer activities. Therefore, a stack designed to exploit joint optimizations may outperform a ``team'' of individually optimized protocols.

To give an example, let us consider ad hoc routing, which is responsible for finding a route toward a destination in order to forward packets. With reference to the classifications reported in [14] and [15], the main classes of routing protocols are proactive and reactive. While reactive protocols establish routes only toward destinations that are in use, proactive approaches compute all the possible routes, even if they are not (and eventually will never be) in use. Typically, reactive approaches represent the best option: they minimize flooding, computing and maintaining only indispensable routes (even if they incur an initial delay for any new session to a new destination). But what happens when we consider the cross-layer contribution that a routing protocol may introduce in a NeSt framework?

To answer this question, we provide an example of cross-layer interaction between a routing protocol and middleware platform for building overlay networks, where the former contributes exporting the locally collected knowledge of the network topology. Building an overlay network mainly consists of discovering service peers, and establishing and maintaining routes toward them, as they will constitute the backbone of a distributed service. The overlay network is normally constituted by a subset of the network nodes, and a connection between two peers exists when a route in the underlay (or physical) network can be established. The task of building and maintaining an overlay is carried out at the middleware layer, with a cost that is proportional to the dynamics of the physical network. Overlay platforms for the fixed Internet assume no knowledge of the physical topology, and each peer collects information about the overlay structure in a distributed manner. This is possible as the fixed network offers enough stability, in terms of topology, and bandwidth to exchange messages. Of course, similar conditions do not apply for ad hoc environments, where bandwidth is a precious (and scarce) resource and the topology is dynamic. In ad hoc networks, cross-layering can be exploited, offering the information exported by the network routing to the middleware layer. The key idea is that most of the overlay management can be simplified (and eventually avoided) on the basis of already available topology information [16]. In this case, the more information available, the easier the overlay management, and for this reason a proactive routing approach results more appealing. To support this claim, let us look at what is described in [5]. This paper describes a cross-layer interaction between a middleware that builds an overlay for peer-to-peer computing and a Dynamic Source Routing (DSR) at the network layer. In this work the DSR algorithm is forced to maintain valid routes toward the overlay peers, even if these routes are not in use. In other words, a reactive routing is forced to behave proactively, with the additional overhead of reactive control packets. The same cross-layer approach with a proactive protocol would probably represent the best joint optimization.

Another example of joint optimization is the extension of routing to support service discovery. A service discovery protocol works at the middleware layer to find out what kind of services are available in the network. As the dynamics of the ad hoc environment determines frequent changes in both available services and hosting devices, service discovery is of fundamental support. The IETF proposes the Service Location Protocol (SLP) [17] to realize service discovery in both Internet and ad hoc networks. Recently, they also underlined the similarity of the messages exchanged in SLP, with those used in a reactive routing such as the Ad hoc On-demand Distance Vector (AODV) protocol [18]. This proposal discusses an extension of AODV to allow service request/reply messages in conjunction with route request/reply. In this proposal there is a background cross-layer interaction that allows SLP to interface directly with AODV, asking for service related messages, providing local service data and receiving service information coming from other nodes. The proposed joint optimization would work even better in the case of a proactive routing protocol such as DSDV or OLSR (see [15] for details of the protocols). In case of proactive routing the service information regarding the local services offered on each node, could be *piggybacked* on routing control packets, and proactively spread around the network, together with local connectivity information. The service discovery communication could be significantly reduced, at the expense of broadcasting routing control packets a few bytes longer. This optimization would result in a proactive service discovery, where a component like the NeSt supports the exchange of service information from the service discovery protocol, at the middleware, with the routing protocol, and vice versa.

Starting from these considerations, and the lessons from Deliverable D8, during the second year we focused on exploiting cross-layer interactions between routing and middleware layers. To this end, at the network level (see Section 3), i) we introduced a scalable, proactive routing protocol, and ii) we extended link-state mechanism to support service discovery. While at the middleware layer we revised the Pastry platform to optimize its performance by exploiting cross-layer interactions. The result of this is the definition of a middleware platform supporting the same API as Pastry but optimized for operating in the ad hoc environment. This platform has been named *CrossROAD*: *CROSS-layer Ring Overlay for AD hoc networks* (see Section 3).

1.5. NeSt Implementation

In order to implement a first prototype of cross-layer architecture, we focused on a specific example of cross-layer interaction between a proactive routing protocol and a middleware platform aimed at building an optimized overlay network for MANET (CrossROAD). The main idea of this platform is to exploit the cross-layer interaction with a proactive routing protocol in order to collect network topology information and to broadcast services information on the network through the flooding of routing packets. To this aim, we selected an open source implementation of the proactive routing protocol OLSR (Unik-OLSR v.0.4.8 [25]), that we had already tested from the functionality and overhead standpoints, as described in Deliverable D8. OLSR provides a default forwarding algorithm that allows the distribution of OLSR messages of unknown types. A user may want to use the optimized flooding technique in OLSR to flood certain information, routing related or not, to all nodes that know how to handle this message. This particular version of Unik-OLSR allows the development of an internal plug-in for the definition of this additional information.

In the case of CrossROAD, the additional information is represented by services identifiers used to associate to each node the list of services locally provided. In fact, each overlay consists of all nodes providing the same service, and every node (in order to join the overlay) has to know all the nodes providing that service. CrossROAD defines a new Service Discovery protocol that associates a unique identifier to each service, and this information is broadcasted on the network piggybacked in routing packets. In this way, assigning to each node a logical address as the result of the hash function applied to its IP address, CrossROAD can autonomously create and manage the overlay network, without requiring remote connections toward other nodes of the network.

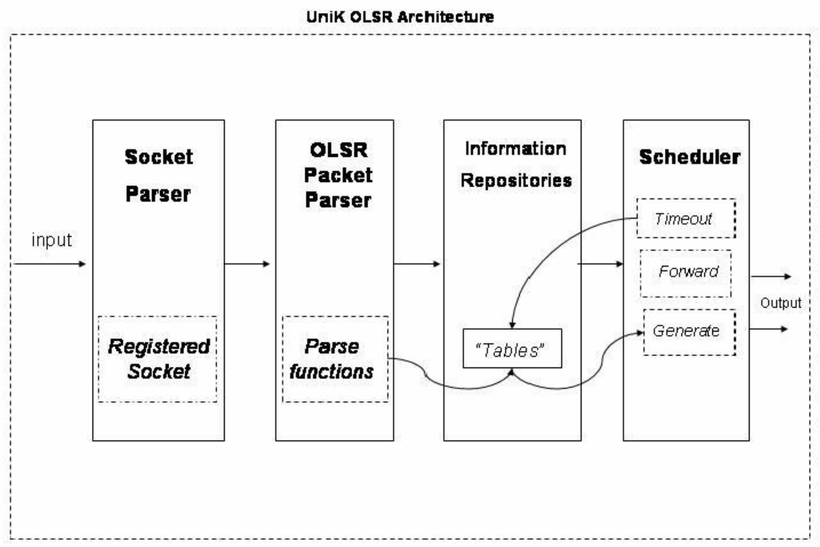


Figure 1.8: Unik-OLSR software architecture

The software architecture of Unik-OLSR is described in Figure 1.8. It consists of four main packages:

- *Socket Parser*: this package waits for incoming traffic on a set of registered sockets. Since it is possible to define additional information to be sent on the network, different software modules may want to interact with OLSR and to do this they have to specify a local socket on which they can communicate. When data is received from one of these sockets, the socket parser calls the function associated with the specified socket. Sockets and their corresponding functions are registered at run-time.
- *Packet Parser*: it receives all incoming OLSR traffic. Particularly it assumes three different behaviours depending on the received packet: it discards the packet if it is found to be invalid; it processes all messages contained in the packet if it recognizes valid message types; it forwards the packet according to the default forwarding algorithm if it does not know the message type. A parse function is associated to each message type in order to process the related messages and update the information repositories.
- *Information repositories*: set of tables where information related to the current state of the network is kept. All calculation of routes and packets are executed based on these repositories. In addition all packet parsing functions update and check the content of these tables to process received messages. The forwarding functionality, in particular, directly accesses to the duplicate table that is a cache of all recent processed and/or forwarded packets. Each entry of these tables is timed out.
- *Scheduler*: it runs different events at different time intervals. To transmit a message at a given interval, one can register a packet generation function with the scheduler. Timing out of tables entries is also triggered by the scheduler. To maintain an information repository that is timed out on a regular period, it is necessary to register a timeout function with the scheduler.

The Unik-OLSR implementation supports loading of dynamically linked libraries (DLL), called *plugins*, to generate and process private message types and any other custom functionality. One of the big advantages of DLLs is that they can be used simultaneously by multiple processes, maintaining only one instance of the library in memory. In this way, plugins provide new functions to an existing application without altering the original application. In addition the definition of plugins does not need to change any code in the OLSR daemon, users are free to implement and license it under whatever terms they like, and they can be written in any language that can be compiled as a dynamic library.

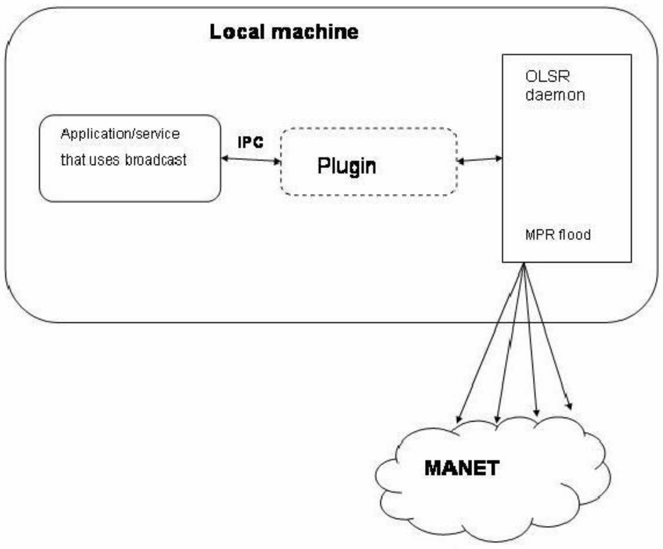


Figure 1.9: Application and routing protocol interaction through the plug-in definition.

Figure 1.9 shows an example of how a plug-in can enable the OLSR daemon to work as a relay for broadcasting application data. In general the software architecture of a plug-in (see Figure 1.10) is mainly represented by its local data structures, and it can be organized in multiple threads in order to manage them and interact with the routing protocol, depending on the purpose of each single plug-in. It communicates with OLSR through the interprocess communication (IPC) using a local socket. When it is loaded by the routing protocol, it has to register the communication socket to the socket parser module of OLSR and to define the new message data structure and the related parsing function. In addition it has a direct interaction with the scheduler module to manage timeouts related to its internal data structures and the generation of additional messages. Finally, it has to define another socket on which it can directly interact with a specific application. Using the IPC model, there are no constraints on the programming languages chosen for the development of the application and the plug-in.

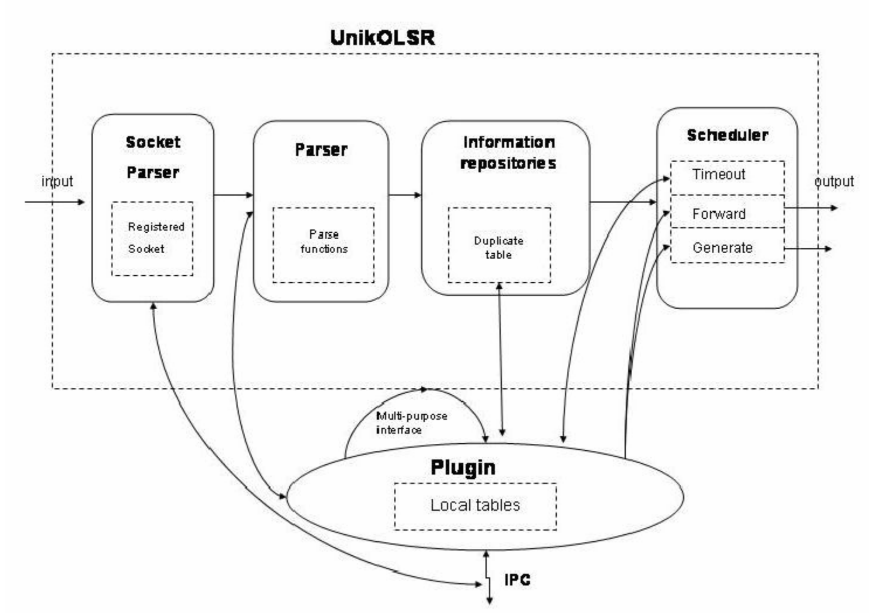


Figure 1.10: Plugin Software Architecture

In our case, the plug-in exactly represents the cross-layer interaction between the routing protocol and the middleware platform (CrossROAD). For this reason it has been called *XL-plugin*. Since each overlay is associated to a single service, multiple instances of CrossROAD can be active on the same node, related to different services. In order to manage all the instances, the XL-plugin has been divided in two main threads:

- *MW-Server*: it registers a local socket to the Socket Parser module to send all additional information. In addition it opens another socket on which it waits for requests from the middleware. When a new instance of CrossROAD is created, it opens a connection to the MW-Server in order to register the service identifier related to the upper-layer application. At this point the MW-Server generates a child thread to manage interactions with each CrossROAD instance. The child thread is responsible for processing CrossROAD messages, updating the local data structures and forwarding the additional information to the routing daemon that will send it on the network through the next packet.
- *Listener*: when the plug-in is loaded, it registers a local socket to the Socket Parser module on which it receives additional information processed by the plug-in parsing function when OLSR receives packets from other nodes.

Figure 1.11 shows how XL-plugin manages interactions between multiple instances of CrossROAD and the routing protocol.

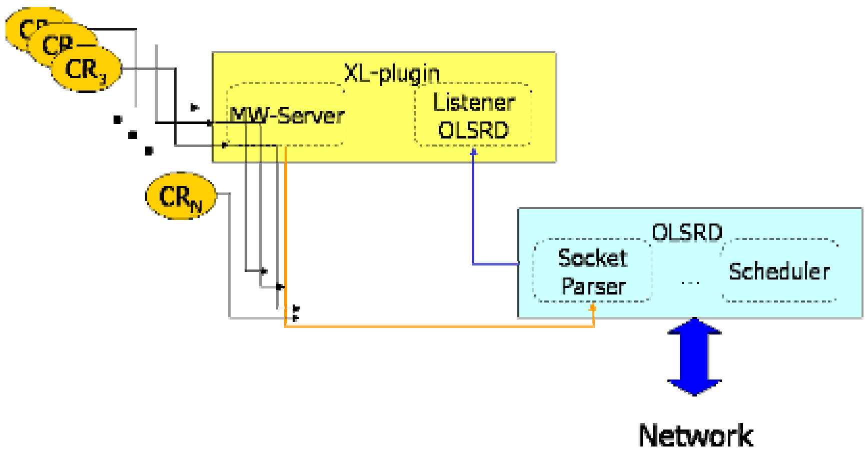
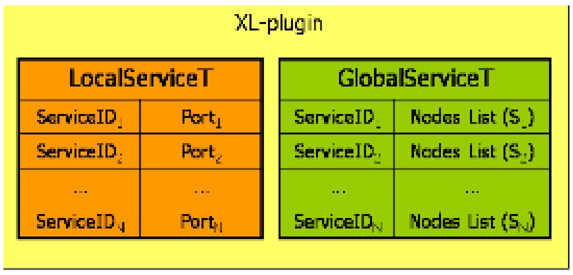


Figure 1.11: XL-plugin as cross-layer interaction between CrossROAD and OLSR

Since XL-plugin has been designed to implement a cross-layer Service Discovery protocol to optimize the overlay management, two main data structures have been defined. In Figure 1.12 they are represented as two tables: *LocalService Table* and *GlobalService Table.* Specifically, the *LocalService Table* maintains the list of services provided by the local node. Each entry of this table consists of a 32-bit service identifier (ServiceID) and the related port number on which it is served by CrossROAD. Instead the *GlobalService Table* maintains, for each service present in the network and currently running on CrossROAD, the list of nodes providing it. For this reason each entry of this table is represented by the service identifier and a dynamic list of elements consisting of the IP address of the related node, and the port number on which the specific service is served. All entries are timed out in order to preserve the consistency of the service information.


XL-plugin

LocalServiceT	
$ServiceID_1$	$Port_1$
$ServiceID_2$	$Port_2$
...	...
$ServiceID_N$	$Port_N$

GlobalServiceT	
$ServiceID_1$	Nodes List (S_1)
$ServiceID_2$	Nodes List (S_2)
...	...
$ServiceID_N$	Nodes List (S_N)

Figure 1.12: XL-plugin internal data structures

More details about interactions between CrossROAD and the plug-in will be explained in Section 5.6.

1.6. Conclusions

Through the definition of a library dynamically loaded by Unik-OLSR implementation of the proactive routing protocol, we developed a first prototype of the cross-layer architecture. Even if it represents only a subset of all cross-layer functionalities presented in Section 1.3, this prototype can be used to validate all advantages of cross-layer interactions in real MANETs. In particular, setting up a real test-bed running OLSR, XL-plugin, CrossROAD and a distributed application developed on top of it, we can declare to have set up a full ad hoc network architecture, focused on middleware and routing interactions, that optimizes the overall system performances.

References

[1] J. P. Macker and M. S. Corson. Mobile Ad Hoc Networking and the IETF. ACM Mobile Computing and Communications Review, 2(3):7–9, 1998.

[2] M. S. Corson, J. P. Macker, and G. H. Cirincione. Internet-based Mobile Ad Hoc Networking. IEEE Internet Computing, 3(4):63–70, 1999.

[3] C. Szyperski. Component Software, pages 140–141. Addison-Wesley, 1998.

[4] K. Chen, S. H. Shah, and K. Nahrstedt. Cross-Layer Design for Data Accessibility in Mobile Ad Hoc Networks. Wireless Personal Communications, 21(1):49–76, 2002.

[5] R. Schollmeier, I. Gruber, and F. Niethammer. Protocol for Peer-to-Peer Networking in Mobile Environments. In Proceedings of 12th IEEE International Conference on Computer Communications and Networks, Dallas, Texas, USA, 2003.

[6] W. H. Yuen, H. Lee, and T. D. Andersen. A Simple and Effective Cross Layer Networking System for Mobile Ad Hoc Networks. In Proceedings of IEEE PIMRC 2002, Lisbon, Portugal, 2002.

[7] M. Chiang. To Layer or not to Layer: Balancing Transport and Physical Layers in Wireless Multihop Networks. In Proceedings of IEEE INFOCOM 2004, Hong Kong, China, 2004.

[8] U. C. Kozat, I. Koutsopoulus, and L. Tassiulas. A Framework for Cross-layer Design of Energy-efficient Communication with QoS Provisioning in Multi-hop Wireless Netwroks. In Proceedings of IEEE INFOCOM 2004, Hong Kong, China, 2004.

[9] V. Kawadia and P. R. Kumar. A Cautionary Perspective on Cross Layer Design. http://black.csl.uiuc.edu/~prkumar/psfiles/cross-layer-design.pdf, 2003.

[10] M. Conti, S. Giordano, G. Maselli, and G. Turi. MobileMAN: Mobile Metropolitan Ad hoc Networks. In Proceedings of the 8th IFIP-TC6 International Conference on Personal Wireless Communications, pages 169–174, Venice, Italy, 2003.

[11] M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross-Layering in Mobile Ad Hoc Network Design. *IEEE Computer*, special issue on Ad Hoc Networks, 37(2):48–51, 2004.

[12] Sun Microsystems. The JAVA Reflection API. http://java.sun.com/j2se/1.4.2/docs/guide/reflection/index.html, 2002.

[13] A. J. Goldsmith and S. B. Wicker. Design Challenges for Energy-Constrained Ad Hoc Wireless Networks. IEEE Wireless Communication, 9(4):8–27, 2002.

[14] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. IEEE Wireless Communications, 6(2):46–55, 1999.

[15] I. Chlamtac, M. Conti, and J. J.-N. Liu. Mobile ad hoc networking: imperatives and challenges. Ad Hoc Networks Journal, 1(1):13–64, 2003.

[16] M. Conti, E. Gregori, and G. Turi. Towards Scalable P2P Computing for Mobile Ad Hoc Networks. In Proceedings of the first International Workshop on Mobile Peer-to-Peer Computing (MP2P'04), in conjunction with IEEE PerCom 2004, Orlando, Florida, USA, 2004.

[17] E. Guttman C., E. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. IETF RFC 2608, June 1999.

[18] R. Koodli and C. E. Perkins. Service Discovery in On-Demand Ad Hoc Networks. Internet Draft, October 2002.

[19] S. Corson. A Triggered Interface. http://www.flarion.com/products/drafts/draft-corson-triggered-00.txt, May 2002.

[20] M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. ACM Computer Communication Review, 33(1):101–106, 2003.

[21] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In Proceedings of the ACM symposium on Communications architectures and protocols, pages 200–208. ACM Press, 1990.
[22] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, 2000.
[23] C. Mascolo, L. Capra, and W. Emmerich. Middleware for Mobile Computing (A Survey). In E. Gregori, G. Anastasi, and S. Basagni, editors, Neworking 2002 Tutorial Papers, LNCS 2497, pages 20–58, 2002.
[24] M. Conti, J. Crowcroft, G. Maselli, T. Turi, "A Modular Cross-layer Architecture for Ad Hoc Networks" in Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, Jie Wu (Editor), CRC Press, New York, 2004 (to appear).
[25] A. Tonnessen, OLSR: Optimized link state routing protocol. Institute for Informatics at the University of Oslo (Norway). Available: http://www.olsr.org

2. WIRELESS TECHNOLOGIES

2.1. Introduction

As motivated in the Deliverable D5 of the Mobile MAN Project, the IEEE 802.11 technology is the most mature and feasible wireless technology to implement real multi-hop ad hoc networks. However, several research papers have highlighted that the ad hoc features of the IEEE 802.11 standard cannot effectively prevent the occurrence of hidden and exposed node problems [WJCD00, WWC01, D01, XGB02, and FZLZG03]. Specifically, to mitigate the impact of the hidden terminal problem on the network performance, the IEEE 802.11 standard introduces a *virtual carrier sensing* mechanism: the handshake of the RTS and CTS control frames between the sender and the receiver before a data transmission. The RTS/CTS access method is aimed at enabling the stations, which are within the transmission ranges of either the sender or the receiver, to estimate the time the channel will be occupied by the ongoing data transmission. However, the RTS/CTS mechanism and its variants assume that all the hidden nodes are within the transmission ranges of the receivers, such that they can correctly receive the CTS packet [FCLA97]. Recent studies conducted through simulations have pointed out that this assumption doesn't always hold in IEEE 802.11-based wireless networks, since some stations can be out of the transmission range of both the transmitting and receiving nodes, but still capable of interfering the frames' reception. The fact that the interference range may be larger than the transmission range has been identified as one of the major reasons of both inefficiency for the RTS/CTS mechanism [XGB03] and TCP unfairness/capture problems [XuS01, XuS02]. Furthermore, simulation and analytical studies [B00, CCG00] have pointed out that the standard 802.11 backoff algorithm significantly degrades the channel utilization in conditions of high-contention, because this policy has to pay the cost of collisions to increase the backoff time when the network is congested. The analytical study of the 802.11 MAC protocol show that, depending on the network configuration, the standard protocol can operate very far from the maximum protocol capacity. Therefore, a number of extensions to the standard backoff protocol have been proposed to improve its performance ([B00, BCD00, BCG04, BCG02,CCG00a]) by adapting the backoff time to the current network contention level in such a way to control the number of stations that transmit in the same slot. Specifically, the authors in [B00, CCG00a] proposed to dynamically control the network congestion by investigating the number of users in the system. However, the knowledge of the number of competing stations is difficult to obtain, and subject to significant errors, especially in bursty arrival scenarios, and in congested systems. On the other hand, the authors in [BCG04] and [BCG02] have exploited recent analytical results, which prove that the average backoff value that maximizes the channel utilization is almost independent of the network configuration (number of competing stations) [BCG03]. Therefore, the maximum channel utilization can be obtained without any knowledge of the number of active stations. Specifically, in [BCG02] the optimal tuning of the backoff algorithm is achieved by controlling the duration of average idle periods and collisions. Instead, in [BCG04] the *Asymptotically Optimal Backoff* (*AOB*) mechanism was proposed, which tunes the backoff parameters in such a way as to avoid that the network contention level exceeds its optimal value. The AOB scheme estimates the network contention level through the measure of the utilization rate of slots. The AOB algorithm was already presented in the Deliverable D5 of the MobileMAN Project, and was indicated as one of the most effective and suitable backoff-tuning algorithm for the 802.*11b*-based wireless environment. However, the AOB mechanism as presented in [BCG04], doesn't consider some critical aspects that have to be solved before adopting it as the backoff tuning algorithm for the *enhanced* IEEE 802.11 wireless card that is under development in the framework of this project. Specifically, the most important open problems are the following:

1. How the AOB mechanism behaves in heterogeneous WLANs where *enhanced* wireless cards should compete with *standard* wireless card for the channel access?
2. How the AOB mechanism behaves in multi-hop 802.11*b*-based ad hoc networks where the partial radio visibility between the stations cause the hidden and exposed node problems, which cannot be effectively solved using the *virtual* carrier sensing, i.e., the RTS/CTS handshake?

All the solutions that have been proposed so far to improve the channel utilization of the IEEE 802.11 protocol were evaluated in network scenarios where every station employs the modified backoff algorithm. To the best of our knowledge, there aren't studies that investigate the performance of these mechanisms in heterogeneous wireless networks, i.e., networks formed by both *enhanced* stations employing the 802.11 MAC protocol extended with an additional contention control mechanism and backoff tuning algorithm, and *legacy* stations adopting the standard 802.11 MAC protocol. Moreover, the backoff tuning algorithm proposed in [BCG04] aimed at guaranteeing that the 802.11 MAC protocol operates close to its theoretical throughput limit, by assuming that there is a complete radio visibility among the stations in the wireless network. However, both assuming homogeneous networks where all the stations adopts *enhanced* wireless cards and considering only single-hop ad hoc networks isn't appropriate for several realistic scenarios. Recently, several small-scale 802.11-based ad hoc networks have been developed to test the proposed communication and networking protocols in real environments. In Section 3 of this deliverable we present a specific test bed for experimenting MANET IETF and novel routing protocols for ad hoc networks. Generally, these real test beds have confirmed that ad hoc networks are intrinsically open environments, where access cannot be limited to specific implementations of wireless network cards. Furthermore, even if the physical carrier sensing range is very large (compared to the transmission range), and it could contain most of the stations around a transmitting one, the obstacles typically present in indoor environments don't allow a complete radio visibility among wireless stations. This section is therefore devoted to an in-depth investigation of the performance of the AOB mechanism in realistic scenarios not considered in [BCG04], and to design and evaluate solutions to make this backoff tuning algorithm more robust and effective.
For ease of reading, and to better understand the basic ideas behind our proposed enhancements, in the following we briefly outline the operations of the AOB mechanism as specified in [BCG04].

2.2. Overview of the AOB algorithm

As explained above, the AOB solution adopts as estimate of the current network contention level the utilization rate of the slots (*slot utilization*), which can be computed as the ratio between the number of slots in the backoff interval[3] in which one or more stations start a transmission attempt, i.e., busy slots, and the total number of slots available for transmission in the backoff interval, i.e., the sum of idle slots and busy slots. The slot utilization definition was originally introduced in [BCD00]. Specifically, [BCD00] proposed the DCC mechanism which exploits the slot utilization (S_U) index for deciding when to perform a transmission attempt or to further defer the access to the channel. Specifically, the DCC mechanism computes a *Probability of Transmission* P_T according to the following formula:

$$P\_T = 1 - S\_U^{N\_A} \qquad (1)$$

where N_A is the number of unsuccessful transmission attempts already performed by the station for the transmission of the current frame. When the standard 802.11 MAC protocol assigns a transmission opportunity to a station (i.e., that station has backoff timer equal to zero and senses the channel idle), the station will perform a real transmission with probability

3 The backoff interval is the time a transmission attempt is deferred due to the random backoff.

P_T; otherwise (i.e., with probability $1-P\_T$) the station deems the transmission opportunity as a *virtual* collision, and the frame transmission is rescheduled as in the case of a real collision, i.e., after selecting a new backoff interval. By using the P_T defined in Equation (1), the DCC mechanism guarantees that the slot utilization of the channel never reaches the value 1.
Numerical results presented in [BCD00] indicate that the DCC mechanism is effective in reducing the contention level, and this is beneficial to increase the channel utilization in 802.11 WLANs. However, DCC operates in a heuristic way, using larger congestion windows than the standard protocol, when the contention increases. The AOB mechanism steps forward because it exploits the analytical characterization of the IEEE 802.11 MAC protocol performed in [CCG00, BCG03] to identify the optimal slot-utilization level the network should obtain to guarantee the maximum channel utilization. Specifically, in [BCG04] it is derived the *Asymptotic Contention Limit* (ACL) function, which approximates the optimal slot utilization with an accuracy that increases with the increase of the network congestion, i.e., the number of active stations. By exploiting the knowledge of the ACL value, the AOB mechanism generalizes the expression of the P_T parameter introduced in the DCC scheme as follows

$$P\_T = 1 - \min\left(1, \frac{S\_U}{ACL}\right)^{N\_A} \tag{2}$$

The P_T expression defined in Equation (2) implies that the probability of performing a transmission attempt tends to zero as the slot utilization approaches the optimal contention level. The AOB scheme guarantees that the system operates asymptotically close to the ACL value, i.e., that the channel utilization is maximal in networks with a large number of stations. It is worth pointing out that the AOB backoff tuning algorithm presents several nice properties that have motivated its adoption as the basic optimization scheme in the *enhanced* 802.11*b* wireless card under development in the framework of the MobileMAN project. Specifically:

1. The analytical characterization of the ACL values presented in [BCG04], proved that the optimal value is almost independent of number of active stations, but depends heavily on the average message length. Hence, the AOB mechanism guarantees to obtain the maximum channel utilization without any knowledge of the number of active stations, overcoming the limitations of previous solutions that adapt the backoff value to the network congestion level [B00, CCG00, and CCG00a].
2. The AOB mechanism tunes the backoff parameters to the network contention level by estimating the network contention level using simple information provided by the standard carrier sensing activity: slot utilization and average size of transmitted frames. Recent measurement studies on IEEE 802.11*b* networks have pointed out that the high-speed wireless channel is characterized by large "interference" ranges [ABCG04]. Therefore, the slot utilization is a robust estimate of the contention level because it is based only on the observation on the channel of the busy and idle periods. It is worth pointing out that the mechanism proposed in [BCG02] is also based on a very simple feedback form the channel: average duration of idle periods and collisions. However, with large interference ranges can be quite problematic in multi-hop wireless networks to discriminate between collisions events and channel occupations. Therefore, the slot utilization provides a more reliable estimate of the channel contention level than the observation of collisions.
3. AOB extends the standard 802.11 access mechanism without requiring any additional hardware.

To conclude this overview of the AOB mechanism, we report numerical results on the effectiveness of the AOB scheme in wireless LANs formed by stations using the high-speed

IEEE 802.11*b* standard. Table 1 lists the parameters' setting used during the simulations. The system parameters are those specified for the 802.11*b* DSSS Physical Layer [IEEEb01].

*Table 1: IEEE 802.11*b *system parameters.*

t_{slot}	*SIFS*	*DIFS*	*EIFS*	Data Rate	CW_{MIN}	CW_{MAX}
20 μs	10 μs	50 μs	364 μs	11 Mbps	32	1024

Figure 2.1 shows the channel utilization of the IEEE 802.11*b* protocol with and without the AOB mechanism, versus the number of wireless stations in the network. These results were obtained by assuming that the stations operate in asymptotic conditions, i.e., that all the stations have always a frame to transmit. Furthermore, we assumed that the message length is constant. In particular, the shown numerical results refer to a payload size of 576 bytes, but similar results were obtained with different values and are omitted. It is worth noting that the results presented in Figure 2.1 differ from the results shown in [BCG04]. This can be easily explained by observing that: *i)* in our experiments we considered the 802.11*b* technology that introduces higher overheads than the low-speed 802.11 technology used in [BCG04], resulting in lower maximum channel utilization; and *ii)* in [BCG04] the CW_{MIN} value was set to 16 slot times, while we adopted the standard value of 32 slot times.

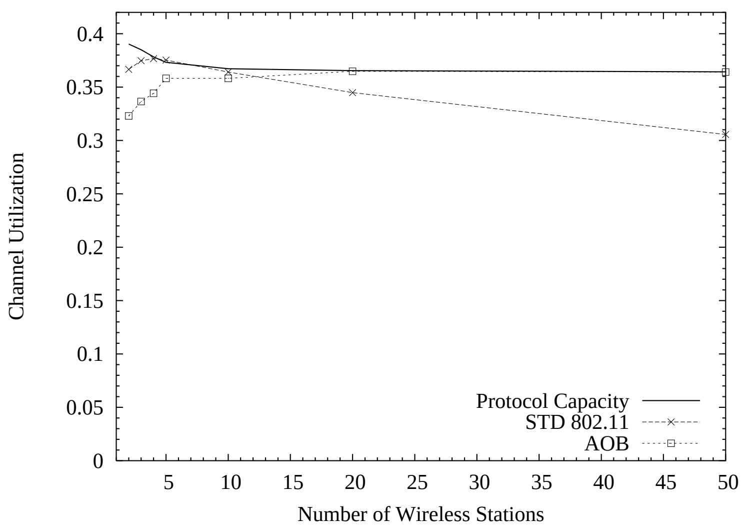


Figure 2.1. Channel utilization of the IEEE 802.11 protocol with and without the AOB mechanism versus optimal value.

2.3. Open Issues

In this section we extensively elaborate on the limitations of the AOB mechanism that have to be overcome in order to adopt this technique as the backoff tuning algorithm for the *enhanced* IEEE 802.11 wireless card that is under development in the framework of this project. In particular, we show that:

- Wireless stations employing the AOB mechanism to maximize the channel utilization, have serious problems when competing with stations adopting the standard binary exponential backoff procedure. The reason of this drawback of the AOB protocol is that this contention control mechanism aims at enforcing a maximum contention level in the network; therefore it is vulnerable to the presence of stations that either are allowed to exceed this limit or to select backoff values that not adequately reflect the current network contention.
- The slot utilization estimate is a concise and *aggregate* characterization of the channel conditions, which can be measured by all the stations inside the physical

carrier sensing range of the transmitting one. However, in multi-hop wireless networks each station has a limited knowledge of the ongoing transmissions, because it can detect only what happens within its radio visibility range. Measurements studies conducted on small-scale 802.11*b*-based ad hoc networks [ABCG04] indicate that the network is affected by a high unfairness, and each traffic flow is advantaged on the other according to the distance between senders and receivers. The AOB mechanism therefore cannot solve these unfairness problems, because the slot utilization estimate as defined in [BCG04] assumes that each station contributes independently and equally to the channel utilization.

2.4. Limitations of the AOB Mechanism in Heterogeneous WLANs

In this section we investigate how the AOB mechanism works in heterogeneous wireless networks, i.e., networks formed by both *enhanced* stations employing the 802.11 MAC protocol extended with the AOB mechanism, and *legacy* stations adopting the standard 802.11 MAC protocol. Although this discussion is specifically focused on the AOB scheme, it could be straightforwardly generalized to any mechanism that aims at improving the channel utilization in 802.11-based WLANs by tuning the backoff value to approximate the *optimal* network contention level. In particular, we show that stations that time-spread their accesses to channel in such a way as to maximize the channel utilization, have serious problems when competing with stations adopting the standard binary exponential backoff procedure. To clearly point out these problems, we have initially considered a network scenario where a single *legacy* wireless station competes with a number n of *enhanced* wireless stations. Figure 2.2 compares the throughput achieved by the *legacy* wireless station (the curve labeled "STD") and by one of the *enhanced* wireless stations (the curve labeled "ENH").

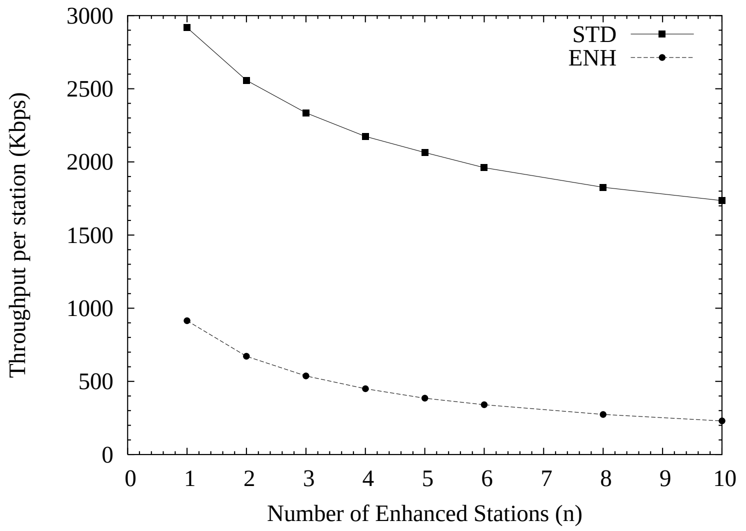


Figure 2.2. Comparison of stations' throughput in a network with one *legacy* station and n *enhanced* stations.

The shown results indicate that the *enhanced* stations are significantly disadvantaged by the presence of even a single *legacy* station: the *legacy* station's throughput is from three (for $n = 1$) to seven times (for $n = 10$) higher than the *enhanced* stations' throughput. This behavior can be explained by observing that the *legacy* station uses an average backoff that depends only on the number of collisions experienced during the frame transmission, while the *enhanced* stations will use larger backoff values because the AOB mechanism introduces *virtual* collisions according to the measured slot utilization. Specifically, even in the case the slot utilization is lower than the ACL value, the AOB mechanism will release transmission opportunities in a probabilistic way, following the expression (2). This implies that the *enhanced* stations will make available to the *legacy* stations channel time otherwise occupied by transmission attempts.

The results presented so far indicate that the behavior of stations using the AOB mechanism in heterogeneous networks heavily depends on the contribution to the slot utilization produced by the *legacy* stations' transmissions. Hence, in the following we further investigate the slot utilization performance to better clarify the reasons of the vulnerability of the AOB scheme. In particular, Figure 2.3 reports a snapshot of the slot utilization as a function of time in a network with n IEEE 802.11*b* *legacy* stations transmitting 576-bytes long frames, comparing it with the ACL value (which is independent of the n value [BCG04]).

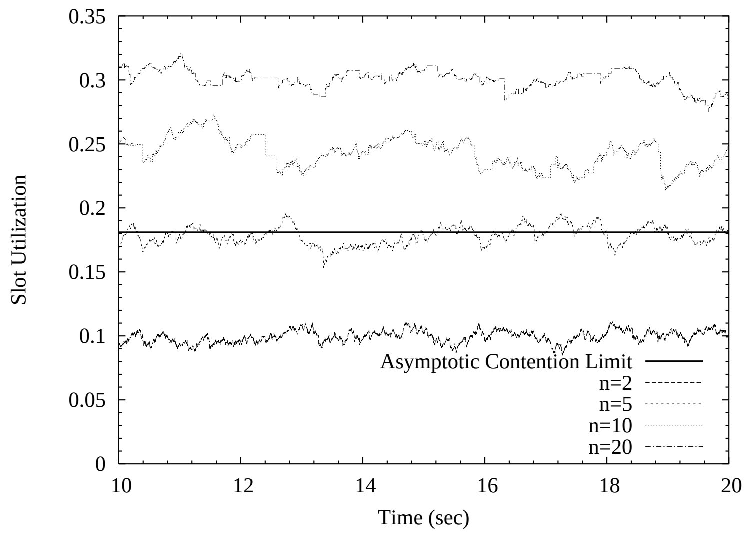


Figure 2.3. Snapshot of the slot utilization as a function of time in a network with M IEEE 802.11 *legacy* stations.

It is straightforward to observe that in conditions of light network loads (i.e., $n \leq 5$), the standard protocol is capable of limiting the slot utilization below the ACL threshold, while for heavier network loads ($n > 5$), the standard backoff procedure fails in guaranteeing an efficient spreading of the channel accesses, causing the increase of the utilization rate of the slots over the optimal limit. On the other hand, the AOB mechanism attempts to ensure that the slot utilization never reaches the ACL threshold by blocking the transmission attempts that could lead to an undue slot utilization level. As a consequence, the higher the number of *legacy* stations, the greater it will be the slot utilization and the lower the probability that the *enhanced* stations could succeed in accessing the channel.

The results reported in Figure 2.3 are useful also to explain why the AOB mechanism performs worse than the standard protocol for small network populations, as shown in Figure 2.1. In particular, we observe that the standard backoff procedure adopts too large backoff values for small network populations, resulting in a waste of channel time. Hence, the channel utilization can be increased only by reducing the time the stations are idle due to the backoff deferral. However, the AOB mechanism is only capable of slowing down the utilization rate of slots, resulting in a further reduction of the channel utilization.

2.5. Limitations of the AOB Mechanism in Multi-Hop Ad Hoc Networks

To better explain the limitations of the AOB mechanism in multi-hop ad hoc networks, initially we briefly outline the most relevant results shown in [ABCG04] investigating the behavior of the 802.11*b* wireless channel. This experimental study has pointed out that the channel model is more complex than usually assumed, and from these results in [ABCG04] an accurate channel model was derived for the high-speed 802.11*b* technology. In this

section, we exploit this model: *i)* to provide explanations of the significant unfairness that affects even very simple network scenarios; and *ii)* to motivate why the AOB mechanism is ineffective in reducing this unfairness.
Traditionally, to characterize how the radio signal affects the receiving stations in low-speed 802.11-based networks, three different regions around a radio source that are discriminated according to the radio propagation properties of the wireless medium:

1. ***Transmission Range*** (R_{tx}) is the range within which a transmitted frame can be successfully received. It is mainly determined by the transmission power.
2. ***Physical Carrier Sensing Range*** (R_{pcs}) is the range within which a transmission can be detected. It mainly depends on the sensitivity of the receiver.
3. ***Interference Range*** (R_{if}) is the range within which stations in receive mode will be "interfered with" by a transmitter, and thus suffer a loss. It depends on the ratio between the power of the received *correct* signal and the power of the received *interfering* signal. It is very difficult to predict the R_{if} because it is a function of the distance between the sender and receiver, and of the path loss model.

Measurements conducted in [ABCG04] show that the channel model is more complex than usually assumed, and more than three regions should be discriminated around a radio source. Specifically, the experimental results indicate that:

- A frame can be correctly received at a distance from the sender that is highly dependent on the bit rate used for transmitting it. Since different bit rates are used for control and data frames, and even for header and payload transmissions, different transmission ranges exist at the same time in the wireless network.
- Transmission ranges for high bit rates are in practice much shorter than usually assumed for the 802.11 technology. The large R_{pcs} with respect to the R_{tx} (the R_{pcs} could be more than 5 times the R_{tx} for the 11 Mbps rate) implies that the dependencies between the stations extend far beyond the R_{tx}, and hidden and exposed problems affect the network performance far beyond the transmission ranges of senders and receivers.

These results indicate that it is a rough approximation to use a single R_{tx} value when modeling the 802.11*b* wireless channel. Hence, assuming that a station S is transmitting at rate r ($r \in \{2,5.5,11\}$ Mbps) towards a destination R located at a distance d from S, the other stations around S can be associated to three different classes:

1. Stations at a distance $d < R_{tx}(r)$ are able to correctly decode the signal transmitted by S, if S is transmitting at a rate lower or equal to r.
2. Stations at a distance $R_{tx}(r) \le d < R_{pcs}$, cannot correctly decode the signal transmitted by S. However, they observe the channel busy and thus they defer their transmissions.
3. Stations at a distance $d > R_{pcs}$ do not measure any significant energy on the channel when S is transmitting, therefore they can start transmitting contemporarily to S. However, some interference due to the R's transmissions may occur depending on the R_{if} value.

Taking in consideration the above channel model, it is straightforward to observe that the behavior of the MAC protocol could be difficult to interpret even in simple network scenarios. To better highlight this aspect, let us consider the basic network configuration depicted in Figure 2.4. The experimental results we describe in the following are obtained locating the four portable computers in an *outdoor* space with no obstacles that could interfere with the radio propagation. The test bed was based on laptops running the Linux-Mandrake 8.2 operating system and equipped with D-LinkAir DWL-650 cards using the 802.11*b* DSSS Physical Layer (PHY). The traffic was generated by UDP flows. If not otherwise stated, each packet is 512 bytes long.

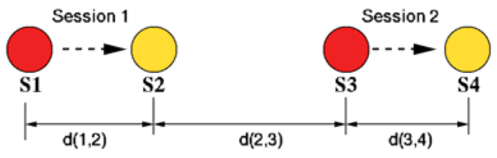


Figure 2.4. Reference network scenario.

In Figure 2.5 we show the throughput per session as a function of the $d(2,3)$ value. The results indicate that the system is affected by a significant unfairness: one session is advantaged with respect to the other according to the $d(2,3)$ value. What is remarkable is that the fairness characteristics are variable: the same session is not always advantaged but distance thresholds exist that cause *inversions* of the fairness properties. Our claim is that there are several factors in 802.11*b* multi-hop ad hoc networks that may introduce unfairness and each of them becomes dominant depending on the distances between transmitters. However, it is difficult to derive exhaustive explanations of these results using only the measurements because the commercial wireless cards usually don't allow tracing fundamental MAC parameters as the backoff timers or the collision events. Hence, the simulations are still fundamental means to gather a complete understanding of the reasons of these phenomena. In Section 2.8.1 we describe the simulation tool that has been developed during the second year of the MobileMAN project to investigate the performance of the 802.11*b* MAC protocol with the realistic channel model described in [ABCG04] and briefly outlined in this section.

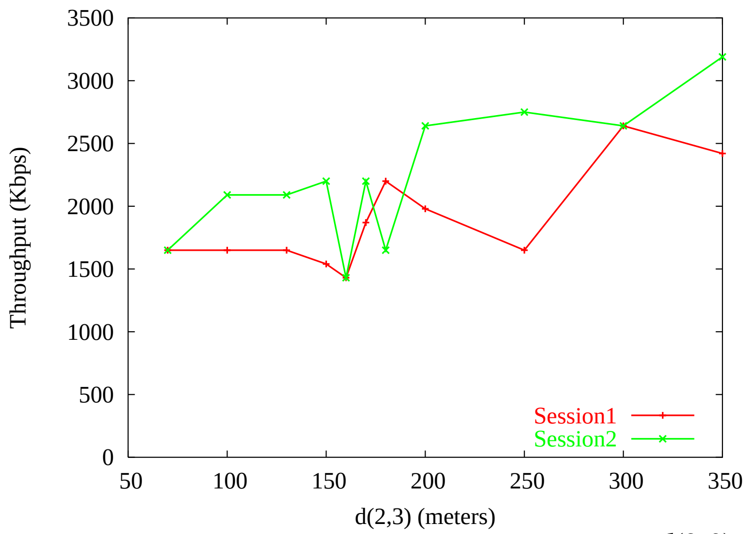


Figure 2.5. Throughput per session as a function of the distance $d(2,3)$ in the network configuration depicted in Figure 2.4.

The results shown in Figure 2.5 clearly indicate that the virtual carrier sensing, i.e., the RTS/CTS access scheme, is ineffective in solving the unfairness issues in 802.11*b*-based ad hoc networks. Specifically, the information carried in the RTS/CTS frames is correctly detectable only within the transmission ranges of senders and receivers, while the channel access is regulated according to the contention level perceived within the *entire* carrier sensing range.

According to the AOB mechanism, each station observes on the channel the idle and busy periods, and exploits this information to control the channel access. However, the slot utilization estimate is an *aggregate* characterization of the network contention level, and the AOB algorithm implicitly assumes that each station contributes independently and equally to the channel utilization. Specifically, each station attempts to reduce the contention level in the network when necessary, releasing randomly transmission opportunities granted by the standard MAC protocol. Stations measuring the same slot utilization have the same probability to skip a transmission attempt. It is straightforward to deduce that the AOB mechanism cannot reduce the system unfairness, because the slot utilization hides totally the unfair contribution of the different transmitters to the channel occupation.

2.6. Enforcing Fairness in Heterogeneous 802.11 WLANs

Results presented in Section 2.4 clearly indicate that any contention control mechanism that operates to guarantee that the network doesn't exceed a maximum *optimal* contention level is vulnerable to the presence of stations that either are allowed to exceed this limit or to select backoff values that not adequately reflect the current network contention. In fact, to regulate the contention level, the *enhanced* stations release transmission opportunities making available channel bandwidth to the other stations in the network, which is unfairly occupied by the stations that don't employ the additional contention control mechanism. Our approach to solve this problem is to design a component capable of estimating the amount of channel time the *enhanced* station is releasing, and exploiting this information to enable the *enhanced* station to recuperate its released time when new transmission opportunities occur.

Although applied in a different context, our approach is similar to the one originally proposed in [BH03, CGKO04, ZCY03] to stimulate the packet forwarding in ad hoc networks. Specifically, the authors in [BH03] introduced a counter, called *nuglet counter*, which is decreased when the node wants to send a packet as originator, and increased when the node forwards a packet. This counter is used to estimate the amount of service the node receives from and provides to the other nodes in the network. In other terms, the nuglets are a sort of virtual currency that is earned when the node forwards packets on behalf of other nodes in the network, and that can be spent to ask other nodes to deliver its own traffic. Similarly, in our solution each station earns a given amount of *credits* when it releases a transmission opportunity with respect to the standard basic access mechanism, credits that can be spent to perform additional transmission attempts.

A question that naturally arises is: how many credits an *enhanced* station earns when it releases a transmission opportunity? To answer this question it is worth noting that when a station releases a transmission opportunity it behaves as a collision has occurred, i.e., it reschedules the frame transmission sampling a new backoff interval after doubling the contention window size. As a consequence, the release of a transmission opportunity from an *enhanced* station, adds extra overhead in terms of a further contention phase. Hence, the *enhanced* station should be remunerated by an amount of credits that adequately counts the duration of this further contention phase.

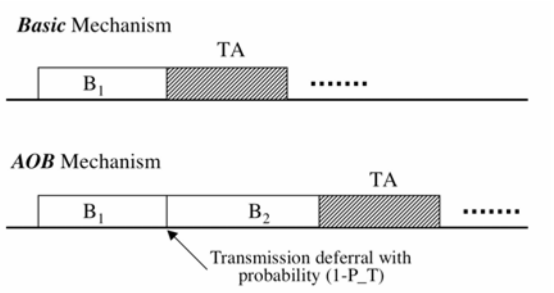


Figure 2.6. Comparison of Basic Access scheme and AOB mechanism behavior.

To better explain this aspect, in Figure 2.6 we compare the behavior of the Basic Access scheme and AOB mechanism. The figure points out that, when the AOB mechanism releases a transmission opportunity, it immediately introduces a new backoff interval B_2. The credits awarded to the *enhanced* station should express the cost of this new backoff. Since, the backoff interval is a random variable, to compute how many credits have to be granted to the *enhanced* station we prefer to consider the contention window used to sample this new backoff interval. Formally, let assume that the *enhanced* station is using the contention window $CW(k) = \min(2^k, 2^{k_{\max}}) \times CW_{MIN}$, where k is the number of the contention phases performed for the current frame transmission[4], and $2^{k_{\max}} \times CW_{MIN}$ is the maximum contention window[5]. Furthermore, let denote with CR_{prev} the credits collected so far by the *enhanced* station. If, according to the AOB mechanism, the *enhanced* station releases the next transmission opportunity, the contention window used to reschedule the frame transmission is doubled up to a maximum value, i.e., $CW(k+1) = \min(2^{k+1}, 2^{k_{\max}}) \times CW_{MIN}$. Hence, the total credits CR_{new} owned by the *enhanced* station are

$$CR_{new} = CR_{prev} + \min(2^{k+1}, 2^{k_{\max}}) \tag{3}$$

So far we have discussed how the stations earn credits. However, a fundamental aspect of our solution is the mechanism used to consume the collected credits. It is worth pointing out that the main effect of the AOB mechanism is to cause the *enhanced* stations to use an average backoff, say $E[B]_{AOB}$, larger than the one used by the standard basic access scheme, say $E[B]_{BA}$. Therefore, to enforce fairness in heterogeneous networks with both *legacy* and *enhanced* stations, the credits should be used by the *enhanced* stations so as to guarantee that they perform their transmission attempts introducing, on average, the same $E[B]_{BA}$ delay of the standard protocol without increasing the contention. To achieve this goal, i.e., reducing the average backoff values used by the *enhanced* stations, we exploit the credits to perform transmission attempts using a null backoff. Specifically, let assume that the *enhanced* station is authorized by the standard backoff rules to access the channel, and it decides to perform a real transmission attempt according to the probability P_T. In this case, our modified mechanism authorizes the station to transmit multiple frames in a burst whether it owns enough credits. It is worth pointing out that transmitting a burst of data frames is equivalent to deliver a longer frame, because we are not introducing new collisions (the concatenated frames are transmitted with null backoff, hence the collision probability is negligible). But, how many credits would be needed to perform an additional frame transmission? To answer this question, it is useful to observe that each successful transmission requires a number of attempts. Therefore, a station for each frame will experience a number of backoff intervals that are sampled from a sequence of contention windows. For instance, let assume that on average two collisions precede a successful frame transmission. In this case the sequence of contention windows used by a *legacy* station would be $\{1,2,4\} \times CW_{MIN}$. This sequence can be straightforwardly translated into a pool of credits by summing up all the window sizes used, i.e., $1+2+4=7$ credits. Generally, the *enhanced* station approximates the average number of credits, say $CR_{TR_{succ}}$, equivalent to the average backoff time spent by a *legacy* station to successfully transmit a frame by counting the number of real collisions experienced, and translating this number into a credits' cost. It is worth pointing out that the *enhanced* station counts only the *real* collisions, ignoring the *virtual* collisions, because these are introduced by the AOB mechanism, but don't correspond to collisions on the channel. Summarizing, when the *enhanced* station completes a transmission attempt, it immediately sends an additional frame with null backoff if $CR \geq \lfloor CR_{TR_{succ}} \rfloor$[6]. In general, the *enhanced* station could send a burst of frames larger than two frames, i.e., formed by a regular

4 The number k is determined by both the real and virtual collisions experienced by the enhanced station.

5 Referring to Table 1, $k_{\max} = 5$ corresponding to a $CW_{MAX} = 1024$ slot times.

6 In general, the $CR_{TR_{succ}}$ value would be a real number, but for the sake of simplicity we approximate it with an integer. The use of the lower integer leads to an underestimate of the cost of immediate transmissions.

transmission plus an immediate transmission. One possible solution is to allow l multiple immediate transmissions if there are enough credits available, i.e., at least $l \times \lfloor CR_{TR_{succ}} \rfloor$. The value of l will determine how aggressively the station consumes its credits. The design of strategies for the dynamic selection of optimal l values is out of the scope of this contribution, therefore in the simulations we used a constant value equal to 2. Finally, it is worth noting a property of correlating the cost of an immediate frame transmission to the number of collisions suffered. Specifically, the more congested is the network, the more are the collisions suffered before a successful transmission and, hence, the higher the $CR_{TR_{succ}}$ value. As a consequence, in congested systems immediate transmission attempts are expensive in terms of credits, while in networks with a few collisions immediate transmission attempts are cheap in terms of credits, and hence are more frequent.
So far we have introduced the core components of our credit-based scheme, i.e., how credits are collected, and how credits are spent. However, there is a final aspect to consider in order integrating our solution in the AOB mechanism. Specifically, the credits can be used to perform additional frame transmissions with null backoff, only after the *enhanced* station has performed a normal transmission attempt following the rules of the AOB mechanism. However, the AOB scheme behaves quite aggressively to reduce the network contention level when the slot utilization is above the ACL value. Specifically, when the slot utilization is higher than the ACL value, the AOB mechanism selects a null P_T value, independently of the N_A value. Figure 2.3 shows that a few *legacy* stations are enough to lead to slot utilization above the optimal value. As a consequence, when the *enhanced* stations compete with *legacy* stations is quite probable that the P_T value is most of the time equal to zero. Hence, the *enhanced* stations earn a lot of credits, but are not allowed to use them.

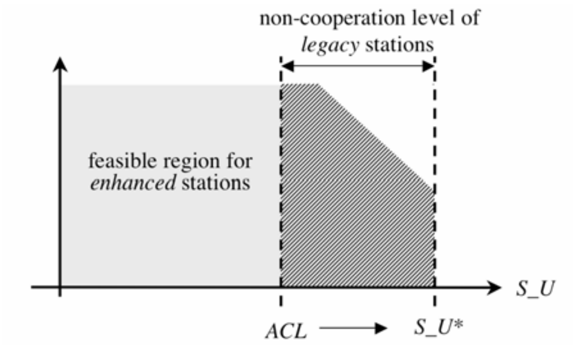


Figure 2.7. How the slot utilization affects the enhanced stations' behavior.

To better explain this aspect, in Figure 2.7 we show how the slot utilization affects the *enhanced* stations' behavior. Specifically, the *enhanced* stations can work only in the region of slot utilization values lower than or equal to the ACL. If the slot utilization observed on the channel, say S_U^*, is greater than the optimal limit, then the *enhanced* stations are blocked. Therefore, the $S\_U^* - ACL$ difference is a measure of the not-cooperation level of the *legacy* stations present in the network. Since the *enhanced* stations cannot force the legacy stations to reduce their slot utilization, there is no reason for refraining from transmitting. Therefore, the *enhanced* stations should enlarge their feasible working region by using a higher ACL value, say ACL^*, equal to or greater than the S_U^*. This could degrade the system efficiency (because we are working above the optimal contention limit ACL), but it ensures fairness between *enhanced* and *legacy* stations
To compute the ACL^* value, we have to design a procedure that allows the *enhanced* stations to become aware that the slot utilization keeps steadily above the ACL. Several mechanisms can be conceived. In the following we propose a quite simple but effective solution that exploits the slot utilization measures which are already available in the basic AOB scheme. Specifically, the AOB scheme computes the slot utilization S_U at the end of each backoff interval, and adopts this value to compute the P_T value according to Equation (2).

However, we cannot directly use the S_U value as an estimate of the cooperation level in the network, information required to decide whether we have to increase the ACL threshold, because the stochastic fluctuations of the slot utilization could lead to instability. In other words, the S_U^* estimate should adopt a much longer temporal scale than the AOB mechanism.

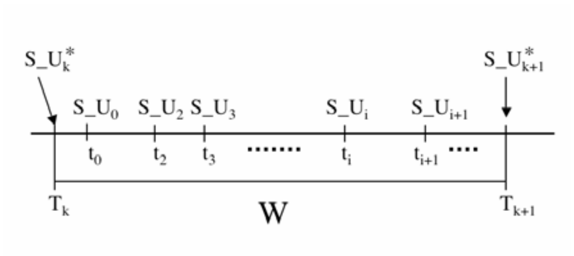


Figure 2.8. Temporal scales used to compute the S_U and S_U^* value.

To better explain this point, it is useful to refer to Figure 2.8. Specifically, the figure shows that the AOB mechanism computes the slot utilization before each transmission attempt, i.e., at the time instant t_i. On the other hand, to estimate the *average* slot utilization S_U^*, the *enhanced* stations observe the network over a constant time window W, which is much longer than the average $\delta t = t_i - t_{i-1}$, i.e., the time between two consecutive transmission attempts. This measure of the S_U^* value should guarantee that the stochastic and transient fluctuations of the slot utilization measures due to the probabilistic behavior of the AOB mechanism are hidden. Therefore, the S_U^* value could be safely used to estimate the contribution to the slot utilization due to the presence of not-cooperative stations. As claimed previously, a measure of how far the system is from the optimal conditions is simply given by $\Delta CL = S\_U^* - ACL$. When $\Delta CL > 0$ the *enhanced* stations could assume that not-cooperative stations are competing with them for the channel access. Therefore, they have to calculate a new contention limit, ACL^*, to avoid to be blocked by the *legacy* stations. Specifically, we define ACL^* as

$$ACL^* = ACL + \max(0, \Delta CL) \cdot \gamma \ , \tag{4}$$

where $\gamma \geq 1$. If we select $\gamma = 1$, then $ACL^* = S\_U^*$. However, two observations motivate the opportunity to use $\gamma > 1$. Firstly, the γ value introduces a guard margin on the new contention limit ACL^*, which reduces the harmful impact of errors on the S_U^* estimate. Secondly, it is beneficial to consider a contention limit greater than the S_U^*, because the *legacy* stations mainly contribute to the S_U^*. Then, overestimating the S_U^* allows the *enhanced* stations to attempt to capture channel time otherwise grabbed by the *legacy* stations. Summarizing, the modified AOB mechanism will use as probability of transmission the following expression

$$P\_T^* = 1 - \min\left(1, \frac{S\_U}{ACL^*}\right)^{N\_A} \ . \tag{5}$$

Finally, it is worth noting that in the P_T^* expression (5) both the S_U and ACL^* values are variable. However, the S_U value is computed at the time instants t_i, while the ACL^* value changes much less frequently, because it is computed at the time instants T_k (see Figure 2.8). Therefore, the ACL^* is a quasi-constant value for the dynamic of the AOB mechanism.

Before concluding this section, it is worth pointing out than many optimizations of the basic credit-based mechanism designed so far are possible, which could exploit the information intrinsically provided by the collected credits. In the following, we introduce two of them, which could be useful to further improve the effectiveness of the proposed solution:

- The standard AOB mechanism requires that after each *virtual* collision, the frame transmission is rescheduled using a doubled congestion window. However, when the number of collected credits is above a given threshold, say CR_{th0}, indicating that the *enhanced* station has frequently back off in the recent past, the station could behave more aggressively. For instance, the *enhanced* station could avoid doubling the contention window. In this way, we have also the positive side effect of not increasing excessively the credits in a station that already owns a large pool of credits. Summarizing, the optimized backoff procedure is the following. Let assume that the *enhanced* station is using the contention window $CW(k) = \min(2^k, 2^{k_{\max}}) \times CW_{MIN}$, where k is the number of unsuccessful transmission attempts already performed for the current frame. If, according to the AOB mechanism, the *enhanced* station releases the next transmission opportunity, then:

$$\begin{aligned} &CW(k+1) = \min(2^{k+1}, 2^{k_{\max}}) \times CW_{MIN} \\ &CR_{new} = CR_{prev} + \min(2^{k+1}, 2^{k_{\max}}) \qquad ; \\ &\qquad (\text{if } CR_{prev} \le \mathrm{CR_{th0}}) \end{aligned} \tag{6}$$

$$\begin{aligned} &CW(k+1) = \min(2^{k}, 2^{k_{\max}}) \times CW_{MIN} \\ &CR_{new} = CR_{prev} + \min(2^{k}, 2^{k_{\max}}) \qquad . \\ &\qquad (\text{if } CR_{prev} > \mathrm{CR_{th0}}) \end{aligned} \tag{7}$$

- In Equations (2) and (5) the N_A parameter is used to assign to stations that have either released several transmission opportunities or performed several unsuccessful attempts, a higher P_T value. However, after a successful transmission the N_A value is reset, and no state information indicating the actual contention level, apart from the estimated slot utilization, is maintained. On the other hand, the owned credits CR are a concise indication of the amount of transmission opportunities released in the recent past, and they should be exploited to give a higher privilege of accessing the channel to stations that more contributed to the slot utilization reduction. Therefore, a quite straightforward extension of the P_T formula is the following:

$$P\_T^{*} = 1 - \min\left(1, \frac{S\_U}{ACL^{*}}\right)^{CR} . \tag{8}$$

In the following performance evaluation of the extended AOB mechanism, which henceforth we indicate as AOB-CR mechanism, both of these two optimizations are adopted.

2.7. Performance Evaluation

In this section, by means of the discrete event simulation, we extensively investigate the performance of the AOB-CR protocol, comparing it with the original AOB scheme that doesn't implement our proposed credit-based mechanism. Table 1 lists the parameters' setting used during the simulations. As far as the AOB-CR-specific parameters: *i)* we use a threshold $CR_{th0} = 20$ credits in the optimization described in formulas (7) and (8); *ii)* the *enhanced* stations are allowed to transmit at most two consecutive immediate frame transmissions, if enough credits are available; and *iii)* $\gamma = 1.1$.

In the following figures, all the curves referring to heterogeneous networks with *legacy* stations and *enhanced* stations using the AOB mechanism are labeled as *AOB*; while all the curves referring to heterogeneous networks with *legacy* stations and *enhanced* stations using the AOB-CR mechanism are labeled as *AOB-CR*. Performance figures have been estimated with the independent replication technique with a 95 percent confidence level. Confidence intervals are not reported into the graphs as they are always very tight (≤ 1 percent).

2.7.1. Heterogeneous Networks

The first network scenario we consider is the same presented initially to highlight the problems of the AOB mechanism in heterogeneous networks, i.e., a network with a single *legacy* station and n *enhanced* stations. Figure 2.9 compares the throughputs achieved by the *legacy* station and by one of the *enhanced* stations when the *enhanced* stations use either the AOB scheme or the AOB-CR mechanism, versus the n value. The results clearly show that the AOB-CR protocol effectively provide a fair channel access to the enhanced stations. In other words, the AOB-CR mechanism guarantees that the *enhanced* stations effectively use the collected credits to reduce the backoff times, so as to introduce, on average, the same contention overheads on their frame transmissions as the basic access scheme. From the numerical results, we can also observe that the enhanced stations are slightly advantaged on the single *legacy* station when n increases. This is due to the fact that the higher the number of enhanced stations in the network, the more transmission attempts are released and the more credits are earned. These credits are used to perform immediate frame transmissions, which suffer negligible collision probability with respect to the normal frame transmissions performed by the *legacy* station.

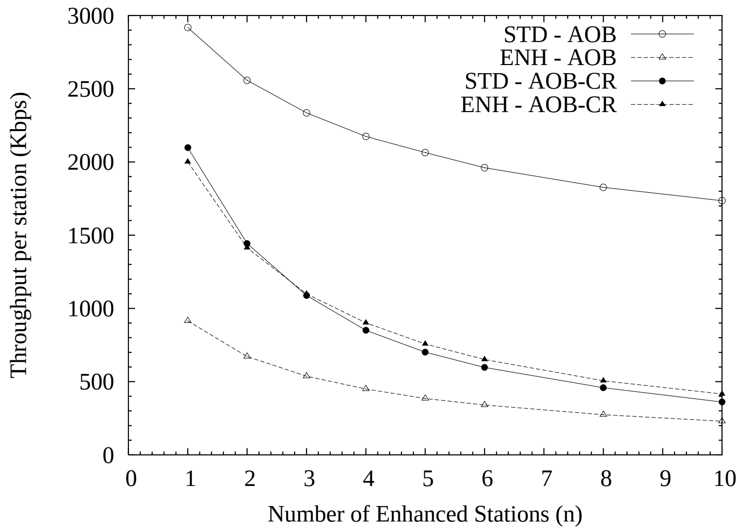


Figure 2.9. Comparison of stations' throughput in a network with one *legacy* station and n *enhanced* stations.

In addition to the case where a single *legacy* station interferes with the operations of several *enhanced* stations, we have investigated more critical network scenarios. Specifically, in Figure 2.10, we compare the throughputs achieved by the *legacy* stations and a single *enhanced* station in a network formed by n *legacy* stations and a single *enhanced* station; while in Figure 2.11 we compare the throughputs achieved by *legacy* and *enhanced* stations in a network formed by n *legacy* and n *enhanced* stations. The numerical results indicate that when there are more than three *legacy* stations, the *enhanced* stations are completely starved if they employ the basic AOB mechanism. On the other hand, the AOB-CR mechanism is effective in correctly estimating the not-cooperation level of the *legacy* stations, so as to use a more appropriate contention limit ACL^*, which avoids that the enhanced stations are starved. Again, as the network population increases, we can notice that the *enhanced* stations obtain a throughout slightly higher than the one achieved by the *legacy* stations.

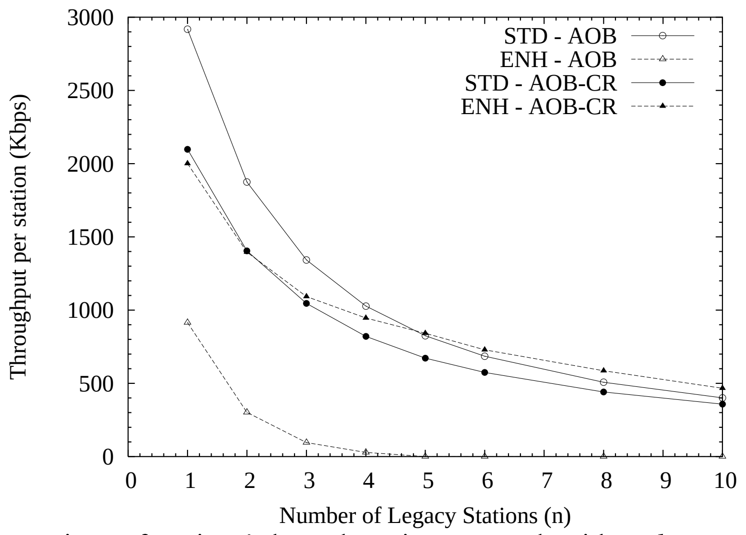


Figure 2.10.Comparison of stations' throughput in a network with n *legacy* stations and one *enhanced* station.

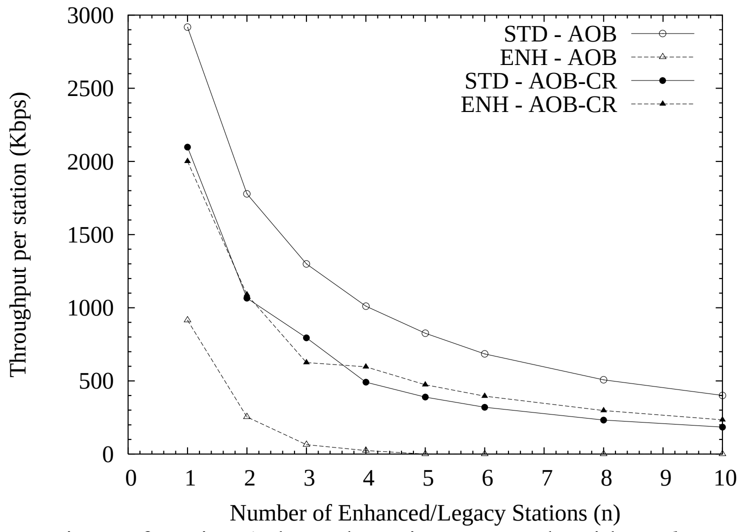
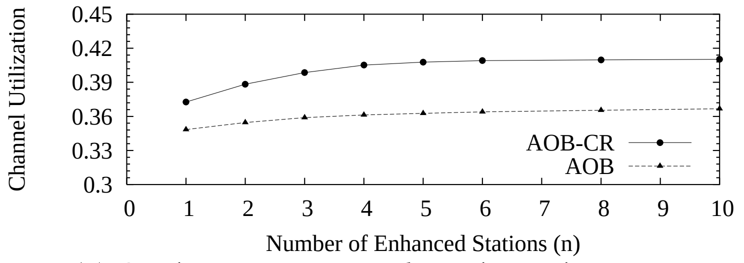
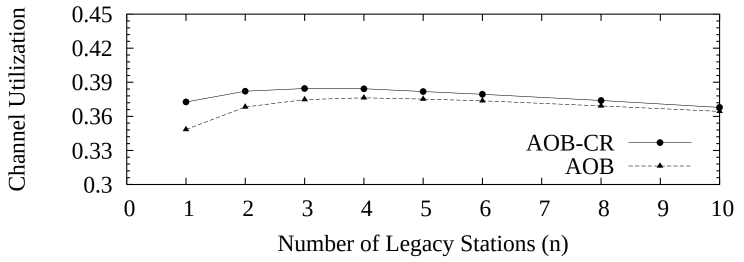


Figure 2.11.Comparison of stations' throughput in a network with n *legacy* stations and n *enhanced* stations.

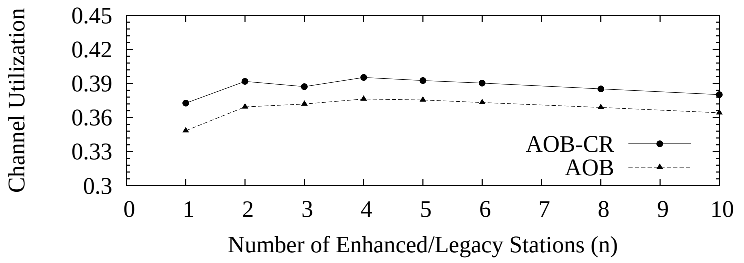
The numerical results reported in the figures above show that the AOB-CR mechanism is effective in enforcing fairness in heterogeneous networks, i.e., in guaranteeing that the *enhanced* and *legacy* stations achieve the same throughput. However, before concluding this study, it is necessary to verify that the fair channel access has not been obtained at the cost of additional protocol overheads, which could results in a reduced overall system efficiency. To this end, in Figure 2.12(a), Figure 2.12 (b) and Figure 2.12 (c) we report the overall channel utilization obtained in the same network scenarios previously studied. The results indicate that the channel utilization is always higher when the *enhanced* stations adopt our proposed scheme instead of the AOB mechanism. This can be explained by observing that the immediate frame transmissions are subject to a negligible collision probability with respect to the normal frame transmissions. Furthermore, the higher is the ratio between the number of *enhanced* and *legacy* stations and the more significant is the gain over the basic AOB scheme.



(a) One legacy *station and n* enhanced *stations.*



(b) n legacy *stations and one* enhanced *station.*



(c) n legacy *stations and n* enhanced *stations.*

Figure 2.12. Comparison of the overall channel utilization achieved when the *enhanced* stations employ either the AOB or the AOB-CR mechanism.

2.7.2. Homogeneous Networks

The results presented in the previous sections indicate that the AOB-CR mechanism solve the problem of guaranteeing a fair coexistence to *enhanced* and *legacy* stations. However, a question that naturally arises is how efficiently the AOB-CR mechanism behaves in a homogeneous network formed only by *enhanced* stations. The AOB-CR protocol operates in such a way to attempt additional frame transmissions when the owned credits allow them. However, these further transmission attempts could increase the slot utilization, preventing the system from operating in optimal conditions. Therefore, in Figure 2.13 we show numerical results comparing the channel utilization in homogeneous networks formed by stations implementing either the basic AOB scheme or the AOB-CR mechanism. The results indicate that our proposed solution outperforms the AOB mechanism for every network population, although it hasn't been specifically designed aiming at this goal. The higher efficiency of the AOB-CR mechanism over the AOB scheme is a positive side effect of the introduction of credits for improving the contention control. In particular, when there are a few stations in the network, the slot utilization maintains below the ACL value, but the credits accumulated during the release of transmission attempts allow immediate frame transmissions. These immediate frame transmissions are characterized by a negligible collision probability; hence they improve the system efficiency.

We can observe that the gain of the AOB-CR mechanism over the AOB protocol decreases by increasing the network population. This can be explained by observing that the more are the stations in the network the higher is the collision probability. Therefore, the immediate frame transmissions become more expensive in terms of credits, and less frequent. However, the efficiency of the AOB-CR scheme cannot be worse than the one of the AOB scheme. In fact, in the case that immediate frame transmissions are infinitely expensive in terms of credits, the AOB-CR behaves similarly as the AOB scheme. Hence, the AOB-CR mechanism tends asymptotically to perform as the AOB scheme.

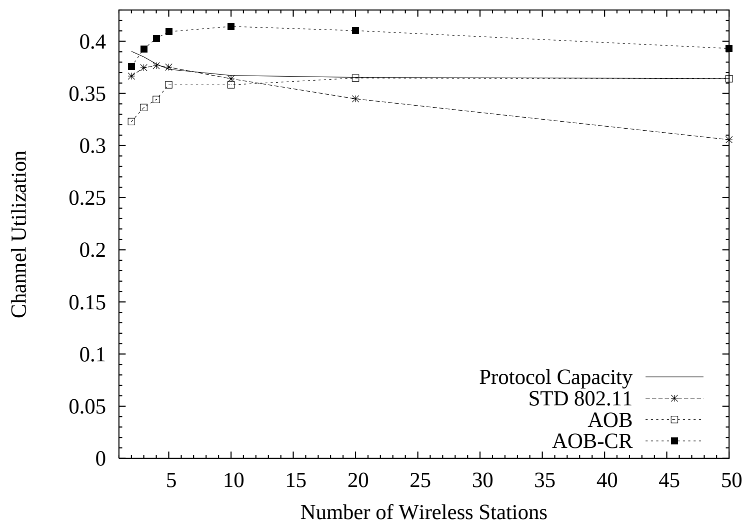


Figure 2.13.Channel utilization of the AOB protocol with and without the proposed credit-based extension versus the standard protocol.

The fact that the AOB-CR mechanism performs efficiently in homogeneous networks is a fundamental property. This implies that the *enhanced* stations can adopt the AOB-CR protocol in any conditions, both in homogeneous and heterogeneous networks, being sure of the improved performance it guarantees. Therefore, it isn't necessary to devise a specific communication protocol that allows the *enhanced* stations to discover the presence of not-cooperative stations in the wireless network, since the AOB-CR scheme is intrinsically capable of adapting to different operating conditions.

2.8. AOB in 802.11 Multi-Hop Ad Hoc Networks

So far we have addressed the problem introduced in Section 2.4, which is the extension of the AOB mechanism to eliminate its vulnerability to the presence of *legacy* wireless stations in the ad hoc network. In the second part of this document we will focus on the issue of making the AOB mechanism effective also in multi-hop ad hoc networks. As discussed in the introduction, the definition of slot utilization as provided in [BCD00] and [BCG04] hides completely the unfair contribution of the different transmitters to the channel occupation in multi-hop ad hoc networks. As a consequence, the AOB mechanism as defined in [BCG04] cannot solve the severe unfairness problems observed in small-scale 802.11*b*-based multi-hop ad hoc networks. In the following, we firstly describe the operations of a simulation tool developed during the second year of this project to investigate the behavior of the MAC protocol in 802.11*b*-based multi-hop ad hoc networks by using the realistic channel model derived in [ABCG04]. This simulation tool has been exploited to derive useful explanations of the unfairness characteristics observed in 802.11*b*-based ad hoc networks, where each traffic flow is advantaged on the other traffic flows according to the distance between senders and receivers. Basing on these observations, we propose a novel definition of slot utilization that could provide a simple and effective way to *quantify* the level of unfairness present in the network. Finally we discuss on how this novel slot utilization index could be exploited to extend the AOB mechanism in order to achieve a *fair* behavior of the MAC protocol, and we provide results confirming the effectiveness of the proposed solutions. Initially, we will show that if the number n of *interfering* stations is known, it is straightforward to extend the AOB technique such that it fairly works in the multi-hop case, by imposing that the ratio between the station's contribution to the slot utilization and the *system* slot utilization is *proportional* to $1/n$. This solution has been tested in the reference network scenario shown in Figure 2.4. However, as extensively discussed in the introduction, it isn't realistic to assume that a station knows how many stations could interfere with its own transmissions. To solve this problem, in this section we design an extension of the AOB mechanism that relies on the *credits* concept introduced in Section 2.6. Specifically, we design: 1) a component capable of estimating the amount of channel time a station is releasing and that could be *unfairly* occupied by other stations in the network; and 2) a mechanism which allows each station to exploit this information in order to reclaim new transmission attempts. Using simulations, we show that the proposed strategy can lead to significant improvements in terms of fair channel allocation, while increasing the channel utilization.

2.8.1. Simulation Tool with a Realistic Channel Model

In this section we aim at describing the simulation tool that has been developed to investigate the performance of the IEEE 802.11*b* MAC protocol and its enhancements considering the realistic channel model for the high-speed wireless channel presented in Section 2.5. As discussed in Section 2.5, the most popular simulation tools [QN03, Ns2] are designed to model the 802.11 technology with channel bandwidth of 2 Mbps. In the simulation studies the following relationship has been generally assumed: $R_{tx} \le R_{if} \le R_{pcs}$. For example, in the ns-*2* simulation tool [Ns2] the R_{tx} and R_{pcs} are set to 250 and 550 m respectively, while the R_{if} is a fixed range as large as 550 m (i.e., maximal interference), which is more than twice of the R_{tx}. A simulation tool that incorporates a more detailed model of the 802.11 physical channel is the QualNet simulator [QN03] the evolution of the GloMoSim simulation library [ZBG98]. Specifically, in this simulation tool R_{tx} and R_{pcs} are set to 367 and 670 m respectively, while the R_{if} is not a fixed range but it is computed by using a *two-ray ground* path loss model. Our work focus on the 802.11*b* technology, thus we decided to use a custom simulation tool since: *i)* we had already a well tested simulator for 802.11*b* single-hop ad hoc networks (see [BCG02] and references herein) easily extendable to the multi-hop case; *ii)* we wanted to have the complete control of the simulation environment in order to fully implement the generalized channel model derived through experimentations in [ABCG04].

To ease the explanation of our results, we briefly outline the operations of the simulator as far as the MAC protocol part. Each station has one transmitting and receiving part, and all the nodes are linked together by a single shared physical channel. When a sender transmits a frame using the bit rate r for the payload, the channel module computes the propagation delay from the sender to every other receiver in the network and passes a copy of the packet to each. When the receiver starts to detect the frame it decides whether it is inside its carrier sense region or receive region by comparing the distance from the sender with the related ranges[7]. If the sender is located outside the carrier sense region, the frame is discarded. If the sender is located inside the carrier sense region but outside the receive region, the PHY passes to the MAC an indication of channel busy, but cannot properly decode the frame. Specifically, the standard mandates that the PHY indicates the beginning of a frame reception only upon the receipt of a valid PHY header. The PHY header has to be transmitted at 1 Mbps (see the standards [IEEE99, IEEEb01] for the details). In general, in the 802.11*b* technology the PHY header is transmitted at 1 Mbps, the MAC header at 2 Mbps, allowing the interoperability with the older 802.11 technology, while the frame payload is transmitted with one of the available bit rates $\{2, 5.5, 11\}$ Mbps. This motivates the presence of multiple transmission ranges in the network. The PHY has to indicate the MAC that a frame transmission is *corrupted* only if a frame transmission was begun that did not result in the correct reception of a complete MAC frame with a correct FCS value. To precisely understand when the MAC is indicated from the PHY of a corrupted frame reception rather than a simple change in the channel state (i.e., from idle to busy, and vice versa) it is important because after the detection of erroneous frames the MAC shall use a larger

7 R_{pcs} and $R_{tx}(r)$ are fixed ranges only affected by the properties of the wireless radios installed at the sender and receiver.

interframe space, the $EIFS$, without regard of the virtual carrier-sense mechanism [IEEE99]. Since in the preliminary phase of our study, we haven't considered frame losses due to the channel noise, a frame can be corrupted only due to the interference of the overlapping reception of other frames. Summarizing, when the PHY starts receiving a frame transmitted at the data rate r, it verifies whether the receiver is idle (see case *a* below) or busy (see case *b* below). The receiver behaves accordingly:

a) The PHY passes to the MAC and indication of channel busy, indication required to properly execute the backoff procedures. The physical header can be correctly decoded only if the distance with the transmitter is lower than $R_{tx}(1)$, while the MAC header can be correctly decoded only if the distance with the transmitter is lower than $R_{tx}(2)$. After a correct reception of the PHY header, the PHY passes to the MAC an indication that has been received a valid start frame delimiter. The frame payload can be correctly received only if the distance with the transmitter is lower than $R_{tx}(r)$. When the MAC frame currently being received is complete, the PHY informs the MAC whether no errors occur during the receive process or the frame was corrupted. Finally, the PHY communicates to the MAC that the channel is idle again.
b) The PHY is already involved in the receive process; therefore this implies that two or more frame reception events partially overlap. Thus, the PHY has to check if the new frame reception can *interfere* with the currently ongoing receptions. Specifically, depending on the R_{if} value, the receiver checks if the sender of the new frame is inside its interference region before deciding that the new frame can corrupt the ongoing reception process. It is worth pointing out that, if the interfering frame disturbs the ongoing frame reception before that the PHY header has been completely decoded, than the receiver doesn't assume that a collision occurred, but only that the channel is busy.

The networks considered in this study are static and we assume that the senders know *a priori* static routes towards the intended destinations. Thus, in our simulations the routing protocol is not a cause of ``interference'', and we have minimized the impact of routing protocol operations on the MAC protocol behavior.

We exploited the simulation tool to conduct and in-depth investigation of the 802.11 MAC protocol behavior in the network scenario depicted in Figure 2.4, which exemplifies the case of a node (S_3) that is *exposed* to the transmissions from a neighbor connection ($S_1 \rightarrow S_2$). The transmission and carrier sense ranges we used are: $R_{tx}(11) = 40$ m, $R_{tx}(2) = 100$ m, $R_{tx}(1) = 110$ m and $R_{pcs} = 160$ m, which conform to the ranges measured in [ABCG04]. As far as the R_{inf} setting we adopted the same approach of the simulator ns-*2* [Ns2], using a fixed range. Since our study is aimed at identifying the impact of the large R_{pcs} over the throughput performance and fairness we assume a minimal interference, i.e., $R_{\text{inf}} = R_{tx}(11)$. In the simulations we used $d(1,2) = d(3,4) = d = 30$ m, such that the distance between the senders and the intended destinations is lower than $R_{tx}(11)$. In Figure 2.14, we investigate and discuss the impact of the distance $d(2,3) = x \cdot d$ on the channel allocation between the two active sessions using both the Basic Access and the RTS/CTS Access. The traffic was generated by UDP flows. If not otherwise stated, each packet is 512 bytes long. All data plotted are computed with a confidence level of 95%.

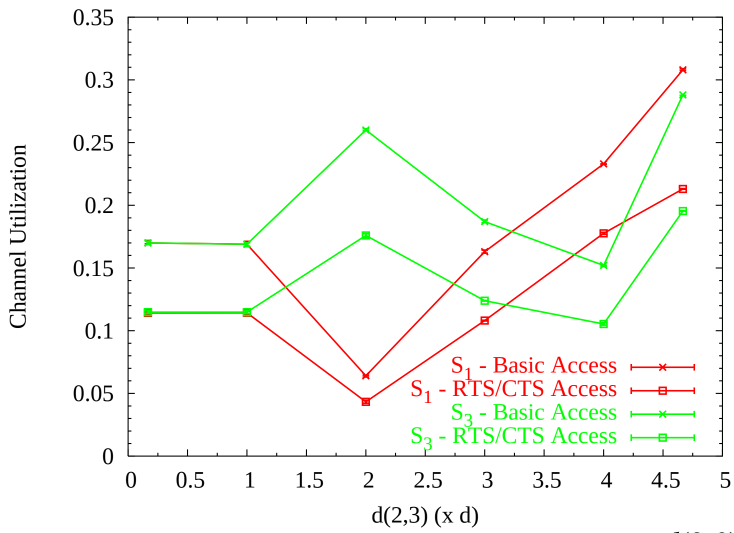


Figure 2.14. Stations' channel utilization as a function of the distance $d(2,3)$ for the network scenario shown in Figure 2.4.

The results shown in Figure 2.14 clearly indicate that the network is affected by a high unfairness, and each session is advantaged on the other according to the distance $d(2,3)$. It is worth noting that the behavior depicted in Figure 2.14 is similar to the behavior shown in Figure 2.5, which was obtained by measurements on real hardware. The differences can be explained by reminding that during the simulations we didn't consider frame losses due to channel noise and we assumed a minimal interference between stations. Both simulations and experiments confirm the high variability of the fairness characteristics in 802.11*b*-based multi-hop ad hoc networks. In the following we provide thorough explanations of the stations' behavior in the case of the Basic Access.

- [$x \le 1$] All the stations can correctly receive the data frames, thus the bandwidth is fairly shared among the traffic sources.
- [$x = 2$] All the stations are within the same carrier sensing area. However, $R_{tx}(11) < d(1,3) < R_{tx}(1)$, thus the transmitters S_1 and S_3 can decode correctly the PHY header recognizing the beginning of the frame transmissions but cannot correctly decode the data payload. This implies that, after the data frame transmission, when the channel becomes idle again they activate an $EIFS$ interval. Since $d(2,3) < R_{tx}(1)$, S_3 correctly decode the ACK frames sent by S_2. The reception of an error-free frame during the $EIFS$ resynchronizes S_3 to the actual busy/idle state of the medium, so the $EIFS$ is terminated and normal medium access (using $DIFS$) continues following the reception of that frame. Unfortunately, $d(1,4) > R_{tx}(1)$ and S_1 cannot correctly receive the ACK frames sent by S_4, thus S_1 starts again an $EIFS$ interval following the indication that the medium is idle after the erroneous ACK frame. Since the $EIFS$ interval is much larger than the $DIFS$ (see Table 1), a S_3's successful access to the channel causes S_1 to defer the transmission for a time longer than the time S_3 has to defer its own transmissions after a S_1's successful access to the channel.
- [$x = 3$] Differently from the case $x = 2$, now $d(1,3) > R_{tx}(1)$, thus transmitter S_1 (S_3) cannot detect the beginning of S_3's (S_1's) frame transmissions, but the PHY still indicates to the MAC that the channel is busy. Therefore, after the channel becomes idle again, both of the transmitters activate a $DIFS$ interval. This implies that S_1 and S_3 decrement their backoff intervals using an almost synchronous timing. The transmitter S_3 is still slightly advantaged over S_1 because it suffers less interference, since $d(1,4) > R_{Inf}$.

- [$x = 4$] Differently from the case $x = 3$, now $d(1,4) > R_{pcs}$, thus transmitter S_1 cannot detect the S_4's ACK frames. On the other hand, S_3 detects both the data frames sent by S_1 and the control frames sent by S_2, therefore the channel appears as busy to S_3 for longer intervals respect to S_1, which can then access the channel more frequently than S_3. This causes the inversion in the fairness characteristics.
- [$x = 4.5$] The distance between transmitters is greater than R_{pcs}, therefore S_1 and S_3 execute the backoff procedures almost independently. Furthermore, $d(2,3) > R_{Inf}$ thus S_3 can begin a data frame transmission in parallel with S_1 without being interfered. S_1 is slightly advantaged with regard to S_3 because when S_2 transmits its ACK frames, S_3 cannot decrements its backoff interval.

Figure 2.14 also shows the channel utilization achieved when using the RTS/CTS mechanism. We obtained the same stations' behavior with a decrease in the observed utilization. It wasn't an unexpected result since we have extensively discussed that the RTS/CTS mechanism wasn't designed to solve the exposed node problem. Moreover, the RTS and CTS frames have the same radio visibility of the ACK frames, therefore they cannot provide further information on the channel allocation and access coordination than the physical carrier sensing.
To investigate the performance in the case of hidden nodes we use the network scenario described in Figure 2.4 by inverting the direction of *session2*, i.e., now stations S_1 and S_4 are the transmitters. It is worth pointing out that this is the same configuration used in [XGB03] to demonstrate the performance degradation of the RTS/CTS mechanism due to a large interference range. However, the authors in [XGB03] considered the 802.11 technology, while in this work we consider the 802.11*b* technology and we focus on the impact of the large carrier sense range. Figure 2.15 shows the simulations results. As expected, the large R_{pcs} with respect to the $R_{tx}(11)$ significantly mitigates the occurrence of the hidden node problem (see also [FCLA97]). The carrier sense activity is sufficient to coordinate the channel access such that to obtain a fair allocation of the channel among the transmitters, because it guarantees that the transmitter S_1 and S_4 have a symmetric view of the channel status. Moreover, the results confirm that the RTS/CTS mechanism cannot increase the channel utilization.

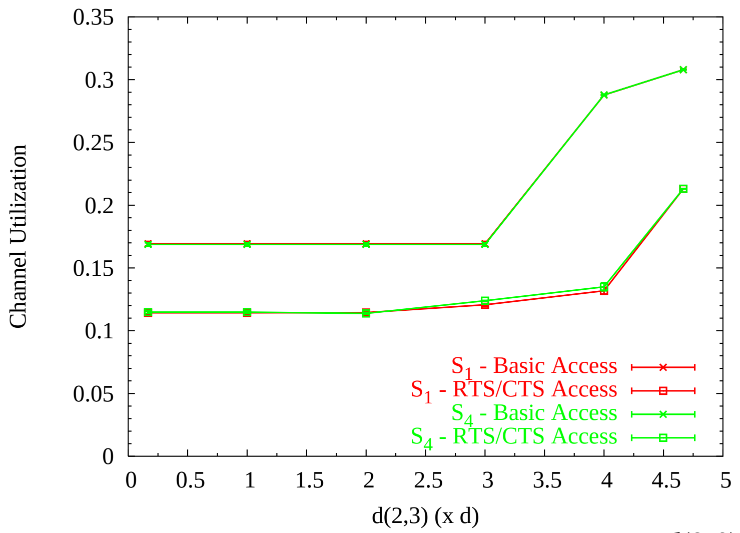


Figure 2.15. Stations' channel utilization as a function of the distance $d(2,3)$ for the network scenario shown in Figure 2.4 with the inversion of the *session2*'s direction.

One of the main objectives of this work is to redesign the AOB mechanism in such a way that it can efficiently operate in multi-hop ad hoc networks, i.e., providing both an optimized and fair allocation of the channel. In the remaining of this section we focus on the exposed node problem because it is the condition that introduces the most remarkable unfairness in the 802.11*b* multi-hop ad hoc networks.

2.9. A Novel Definition of the Slot Utilization

The results presented so far indicate that virtual carrier sensing mechanisms cannot solve the unfair allocation of the channel among the stations, because they fail to fairly estimate the network contention level. In the following, we show that better information on the current network contention level is already available at the MAC layer by exploiting the physical carrier sensing mechanism. Specifically, the rate of utilization of the slots was identified in [BCD00] as an effective estimate of the network congestion level in 802.11 WLANs. The original definition of the slot utilization is the ratio between the number of slots in the backoff interval in which one or more stations start a transmission attempt, i.e., busy slots, and the total number of slots available for transmission in the backoff interval, i.e., the sum of idle slots and busy slots. This information is simple to obtain because it is granted by the standard carrier sensing activity. However, this definition of the slot utilization provides an aggregate measure of the network contention level, hiding the unfair allocation of the channel access. Thus we propose a novel definition of the slot utilization such that each transmitter S_k can differentiate between the contribution to the channel occupation due to its own transmissions and to its neighbors' transmissions. Specifically, selected a fixed observation period T, each transmitter S_k computes an *internal* slot utilization, $S\_U_{Int}^{k}$ and an *external* slot utilization $S\_U_{Ext}^{k}$ in the following way

$$S\_U_{Int}^{k} = \frac{n_{tx}}{n_i + n_{rx} + n_{tx}}, \tag{9}$$

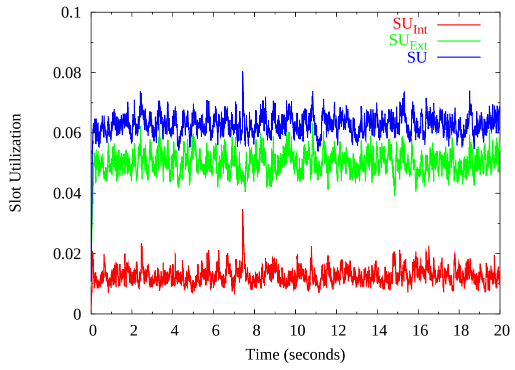
$$S\_U_{Ext}^{k} = \frac{n_{rx}}{n_i + n_{rx} + n_{tx}}, \tag{10}$$

where n_{tx} is the number of transmissions performed by S_k during the interval T; n_{rx} is the number of *separate* channel occupations observed during T; and n_i is the time, normalized to the time slot, the channel was idle during T (including the $DIFS$ and $EIFS$ periods [IEEE99]). A reception event is any indication coming from the PHY that the channel status changed from idle to busy. Two reception events have to be considered separate if the time between the end of the former and the beginning of the latter is greater than a $DIFS$ interval[8]. The aggregate slot utilization, S_U^{k}, is clearly the sum of the $S\_U_{Int}^{k}$ and $S\_U_{Ext}^{k}$ values. It is worth pointing out that in single-hop wireless networks, each station will observe the same slot utilization. On the other hand, in multi-hop wireless networks, the reception events a station observes are dependent on the distance of that station form the transmitters. As a consequence, the ratio between the $S\_U_{Int}^{k}$ and $S\_U_{Ext}^{k}$ values can be exploited by node S_k to evaluate if it retained an unfair portion of the channel. In particular, $S\_U_{Int}^{k} > S\_U_{Ext}^{k}$ indicates that S_k is accessing the channel more frequently than its neighbors. Therefore, before any transmission attempt, S_k should not only control that its backoff counter is equal to zero, but also that its transmission doesn't increase the system unfairness. Before discussing how the AOB mechanism can be tuned in order to achieve a fairer allocation of the channel among the stations, it is useful to verify that formulas (9) and (10) are effective and accurate estimates of the system behavior from the channel allocation's perspective. Hereafter, we use $T = 5$ ms, unless explicitly specified. In Figure 2.16 we show the $S\_U_{Int}^{k}$ and $S\_U_{Ext}^{k}$ for

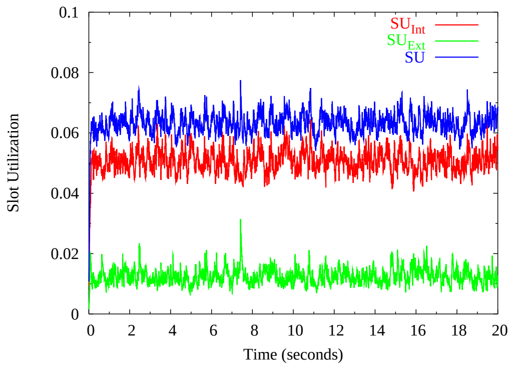
[8] This guarantees that each station counts a frame exchange as a single reception event.

the transmitter S_1 and S_3 at the distance $d(2,3)=60$ m, and in Figure 2.17 for the distance $d(2,3)=120$ m, when the unfairness in the network is more remarkable. To avoid sharply fluctuation in the slot utilization we use a low-pass filter[9] to smooth the measurements' variations.

Firstly, we can observe that the station getting more channel has always a $S\_U_{Int}^k$ greater than the $S\_U_{Ext}^k$. This implies that the slot utilization can effectively indicate the occurrence of an unfair allocation of the channel in the case of exposed nodes. The second important consideration that can be derived from the results is that, when the stations are all in the carrier sense region, as at the distance $d(2,3)=60$ m, the transmitters measure almost reciprocal slot utilization: $S\_U_{Int}^1 \cong S\_U_{Ext}^3$ and $S\_U_{Ext}^1 \cong S\_U_{Int}^3$. This implies that the slot utilization is effective also to *quantify* the unfair level of the channel allocation. Instead, at the distance $d(2,3)=120$ m, the transmitter S_1 cannot detect the S_4's frame transmissions, therefore it will measure, during the observation period T, a larger idle time than S_3. This explains why node S_1 measures slot utilization indexes lower than node S_3. However, even in this case the slot utilization is still a valid indication of the unfair channel allocation between transmitters S_1 and S_3.

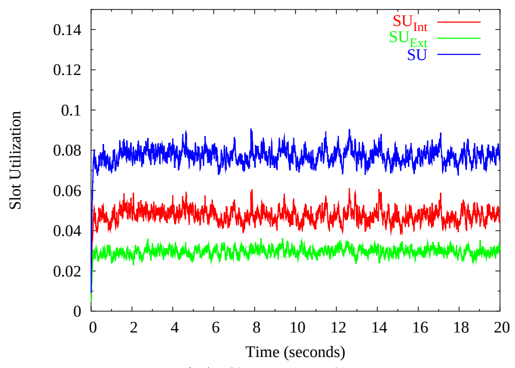


(a) Station S_1

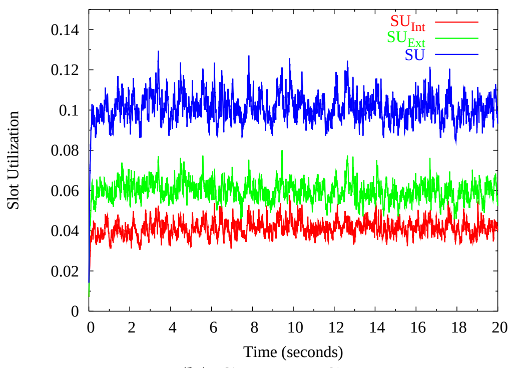


(b) Station S_3

Figure 2.16. Slot Utilizations measured in the networks scenario of Figure 2.4 for $d(2,3)=60$ m.



(a) Station S_1



(b) Station S_3

Figure 2.17. Slot Utilizations measured in the networks scenario of Figure 2.4 for $d(2,3)=120$ m.

9 The low-pass filtering is implemented using a moving average-window estimator of parameter 0.9 [BCG02].

2.10. Tuning the AOB Mechanism to Enforce Fairness

In this section, we propose and evaluate extensions of the original AOB mechanism that could guarantee that the enhanced MAC protocol is fair also in multi-hop ad hoc networks. A fair MAC protocol should guarantee that when the network is saturated, each station is able to obtain a fair share of the channel. A first issue is to decide what is a fair allocation of the channel, i.e., for each station S_k, which is the portion ϕ_k of the channel S_k should get. Generally, the value of ϕ_k should be determined based on the application requirements. However, we focus on the fairness at the MAC layer, i.e., fairness in the contention for the wireless medium, therefore it is reasonable to adopt the notion of *per-station* fairness: each station should achieve the same proportion of the channel bandwidth. In other words, if n is the number of active stations around a transmitter, the ϕ_k value should be $1/(n+1)$. As explained in the introduction, the estimate of the number of active stations is problematic in wireless network, and even more challenging in multi-hop ad hoc networks. Initially, our main objective is to show the feasibility of a fair channel allocation scheme based on the contention control implemented exploiting the slot utilization concept. For this reason, initially we have assumed to know the number n. Furthermore, for the sake of simplicity, we decide to follow the same approach used in [BWK00], i.e., that all the stations are trusted and cooperative. During this study we have considered the network scenario of Figure 2.4 where two single transmitters are contending for the channel resources, and one transmitter is exposed to the other transmitter's communications In this case, each station should operate in such a way to get a share $\phi_k = 0.5$. The condition that each station gets the same channel share as all its neighbors can be straightforwardly translated into requesting the each station S_k measures a $S\_U_{Int}^k$ equal to the $S\_U_{Ext}^k$. As indicated before, the ratio between $S\_U_{Int}^k$ and $S\_U_{Ext}^k$ could be exploited to filter the transmission attempts. When a station deems that a new transmission can worsen the unfairness of the channel allocation, it should defer its transmission attempts. This can be accomplished as in the AOB mechanism [BCG04]: when the backoff algorithm grants a transmission opportunity to the station, it performs an additional control on its $S\_U_{Int}^k / S\_U_{Ext}^k$ ratio to decide whether it should carry out the transmission. Specifically, when the backoff counter is equal to zero and the channel is idle, the station will perform a real transmission according to the algorithm described in Equation (11), otherwise the transmission is rescheduled as a collision would have occurred, i.e., a new backoff interval is sampled using the standard backoff algorithm.

$$\begin{cases} \dfrac{S\_U_{Int}^k}{S\_U_{Ext}^k} \le C_0 & \text{always transmit} \\ C_0 < \dfrac{S\_U_{Int}^k}{S\_U_{Ext}^k} \le C_1 & \text{transmit with probability } P\_T_0 \\ C_1 < \dfrac{S\_U_{Int}^k}{S\_U_{Ext}^k} \le C_2 & \text{transmit with probability } P\_T_1 \\ \dfrac{S\_U_{Int}^k}{S\_U_{Ext}^k} > C_2 & \text{don't transmit} \end{cases} . \qquad (11)$$

where C_0, C_1 and C_2 are all greater than one. The tuning of the decision process is executed by properly selecting the constant thresholds C_0, C_1 and C_2, and the transmission probability P_T_0 and P_T_1. The smaller the C_i values, the more transmission opportunities are skipped. In the experiments we conducted to test the effectiveness of the proposed solution we use the parameter setting listed in Table 2.

Table 2: Parameter setting for the algorithm described in Equation (11).

C_0	C_1	C_2	P_T_0	P_T_1
1.1	1.2	1.3	0.25	0.5

We can note that stations getting a lower channel share, i.e., with $S\_U_{Int}^k / S\_U_{Ext}^k \leq 1$, will continue to access the channel according to the standard protocol, without skipping transmission opportunities or using shorter backoff intervals. This design choice is motivated by the assumption that the stations are cooperative, therefore stations with unfair channel shares will try to correct their behavior. In Figure 2.18 we show the simulation results obtained when our proposed algorithm is used to extend the standard backoff algorithm in the network configuration described in Figure 2.14. Figure 2.18 shows that the transmitters S_1 and S_3 equally share the channel bandwidth by applying our enhanced transmission policy. Moreover, the fair allocation of the channel among the stations is not obtained by reducing the MAC protocol efficiency. In fact, in Figure 2.18 we also compare the aggregate channel utilization achieved by the standard protocol and the modified one, and we observe that they are almost the same. It is worth pointing out the similarities between Equation (11) and Equation (2), which is used in the original AOB mechanism. Specifically, in our proposed enhancement the *Probability of Transmission* P_T has been modified to adopt a step-wise function, which depends on the ratio between $S\_U_{Int}^k$ and $S\_U_{Ext}^k$ instead of the sum of $S\_U_{Int}^k$ and $S\_U_{Ext}^k$ as in Equation (2).

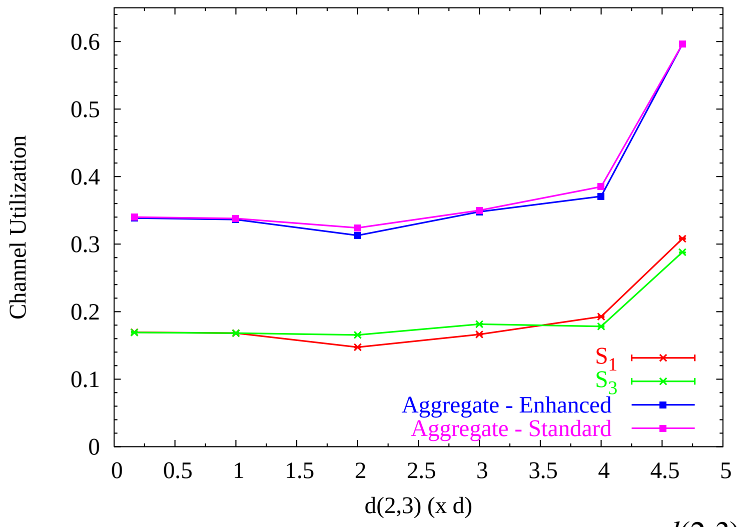


Figure 2.18. Stations' channel utilization as a function of the distance $d(2,3)$ for the network scenario shown in Figure 2.4 using the modified AOB mechanism.

It is clear that the above approach could guarantee the per-station fairness in a generalized network configuration only if the ϕ_k value (i.e., the actual share of the station S_k) is selected taking into consideration the S_k's neighborhood size (i.e., the number n of active nodes around S_k). Specifically, to enforce a $\phi_k = 1/(n+1)$, it is sufficient to guarantee that the $S\_U_{Int}^k / S\_U_{Ext}^k$ ratio is equal to $1/n$. However, it is a demanding challenge to conceive a *distributed* protocol operating at the MAC layer to dynamically estimate, even roughly, the parameter n in wireless multi-hop ad hoc networks. For this reason, in the remaining of this section, we present an extension of the AOB mechanism that doesn't require the knowledge of the parameter n to enforce a fair channel access in multi-hop ad hoc networks. To test our proposed solution we introduce a more complicated network configuration than the one described in Figure 2.4. Specifically, Figure 2.19 shows a network scenario where a number n of transmitters S_i are all exposed to the transmissions of a single station R_1. Obviously, when $n = 1$ this generalized network configuration reduces to the reference network scenario shown in Figure 2.4.

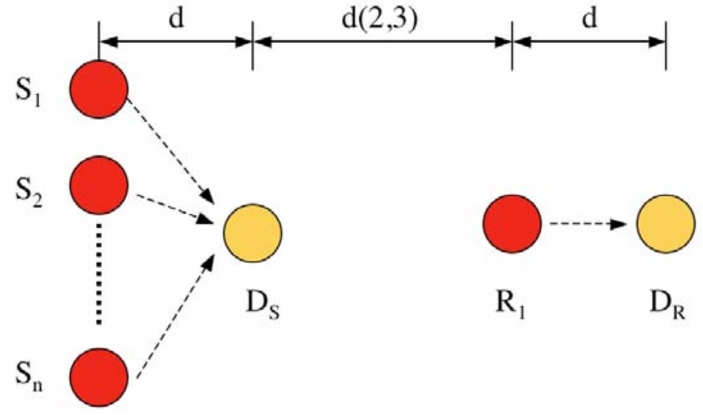


Figure 2.19: Generalized reference network scenario.

The extension of the AOB mechanism we propose to ensure a fair channel allocation also in multi-hop ad hoc networks is based on the same concepts introduced in Section 2.6. Specifically, our approach to solve the unfair problem in multi-hop ad hoc networks is to introduce the *credits* to quantify the amount of channel time that could be unfairly occupied by other stations' transmissions. As explained previously, the original AOB mechanism releases transmission opportunities with respect to the standard basic access mechanism according to the *Probability of Transmission* P_T. Since, when a station releases a transmission opportunity, the credits collected by that station should estimate the amount of channel time the station is releasing. Formally, let assume that the station releases the next transmission opportunity according to the AOB mechanism, and the contention window used to reschedule the frame transmission is CW_{new}. Hence, the total credits CR_{new} owned by the *enhanced* station are

$$CR_{new} = CR_{prev} + \frac{CW_{new} - 1}{2}, \qquad (12)$$

where the term $(CW_{new} - 1)/2$ accounts for the *average* backoff introduced before the rescheduled frame transmission. However, the formula (12) doesn't consider one of the most substantial causes of unfairness in multi-hop networks: the asymmetry in the channel idle times due to the use of the $EIFS$ timer after the detection of erroneous frames [ZNG04]. Specifically, due to the partial visibility among the stations in multi-hop environment, the same frame could be correctly received by one station, say A, triggering a $DIFS$ timer in this station, while it appears as a corrupted frame by another station, say B, triggering an $EIFS$ timer. Since $EIFS >> DIFS$, the deferment of B's transmissions is much longer than the delay introduced before the A's frame transmissions, resulting in unfairness[10]. We claim that the credits owned by a station should take into consideration the amount of channel time the $EIFS$ timer is active, because this idle time could be used by other stations to unfairly decrease their backoff timers. Formally, let denote with $EIFS_{tot}$, the time interval during which the $EIFS$ timer is active. Generally, the $EIFS_{tot}$ value could be either lower or greater than the $EIFS$ value. In fact, the standard mandates that the "reception of an error-free frame during the $EIFS$ resynchronizes the station to the actual busy/idle state of the medium, so the $EIFS$ is terminated and normal medium access (using $DIFS$ and, if necessary, backoff)

10 A more exhaustive discussion on the problems caused by the $EIFS$ timer in 80.11-based multi-hop networks can be found in Section 2.8.1, while a discussion on the problems caused by the $EIFS$ timer in 802.11-based multi-hop networks can be found in [ZhifeiNG04].

continues following reception of that frame" [IEEE99]. On the other hand, the reception of corrupted frames during the $EIFS$ causes the $EIFS$ timer to be reset, because the normal medium access should be reactivated only after a continuous $EIFS$ interval of idle time. To summarize, when the $EIFS$ timer is terminated (either because of the reception of an error-free frame or the expiration of the $EIFS$ timer), the total credits CR_{new} owned by the *enhanced* station are

$$CR_{new} = CR_{prev} + \frac{EIFS_{tot}}{t_{slot}} \quad . \tag{13}$$

Equation (12) and (13) explains how the stations earn credits. However, a fundamental aspect of our solution is the mechanism used to consume the collected credits. Similarly to the extension proposed in Section 2.6, we claim that the stations should use their credits to perform multiple frame transmissions when they are allowed to access the channel. In particular, multiple frames could be transmitted in a burst in order to reduce the overheads and to recover channel time unfairly occupied by other stations in the network. It is worth pointing out that the multiple frame transmission is one of the MAC enhancements that are currently under investigation and standardization within the 802.11*n* Task Group[11] [XR03]. One of the critical aspects that should be considered when introducing the multiple frame transmission capability is the size of the frames' burst. One possible solution is that the number of multiple frames should not be larger than a threshold (such as 2, 3 or 4). However, our scheme could exploit specific information on the actual contention level in the network to define a more sophisticated and efficient strategy to select the burst size. Specifically, our modified mechanism authorizes the station to transmit multiple frames in a burst whether it owns enough credits (i.e., enough released idle slots). To compute how many credits would be needed to perform an additional immediate frame transmission, it is worth noting that each successful transmission could require a number of attempts. As a consequence, a station for each frame will experience a number of backoff intervals that are sampled from a sequence of contention windows. Therefore, each station should maintain an estimate of the average backoff used to successfully transmit the first frame of the burst, say $\overline{B}_{succ}$. When the station succeeds in transmitting a frame according to the rules of the AOB mechanism, it sends a burst of k additional frame with null backoff if $CR \geq k \times \overline{B}_{succ}$. To ensure that the station holds the channel for a reasonable time, we also introduce a threshold l to limit the maximum size of the frames' burst, such that $k \leq l$.
So far we have implicitly assumed that the stations adopt the original AOB mechanism, i.e., they decide to release a transmission opportunity according to the P_T formula (2). However, we have extensively discussed in Section 2.5 that the AOB mechanism is fair only if all the stations in the network measure the same slot utilization, so as to release transmission opportunities using the same probability. As a consequence, in multi-hop ad hoc networks the credit scheme illustrated above couldn't be sufficient to solve all the unfairness problems, but we have to properly modify the P_T formula. Specifically, we propose that each station S_k computes the *Probability of Transmission* according to the following equation:

$$P\_T = 1 - \min\left(1, \frac{S\_U_{Int}^{k} + S\_U_{Ext}^{k}}{ACL - S\_U_{Int}^{k}}\right)^{N\_A} \quad . \tag{14}$$

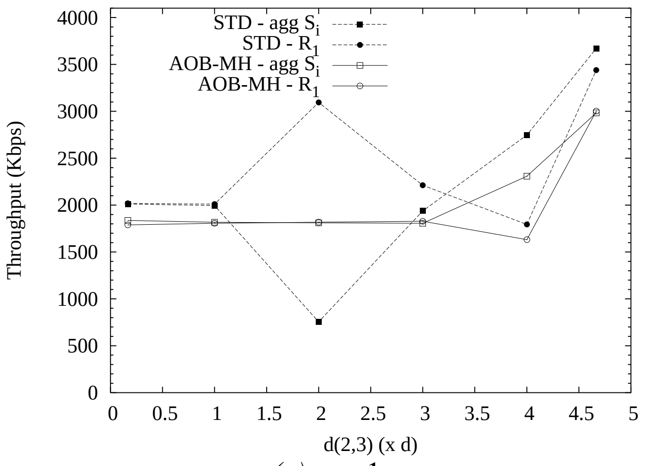
The most significant difference between formula (14) and formula (2) is that the asymptotic contention limit ACL is reduced of the internal slot utilization $S\_U_{Int}^{k}$ measured by the station. The motivation behind this choice is based on the observation that the larger the

11 The goal of IEEE 802.11n is to provide higher throughput via MAC and PHY enhancements.

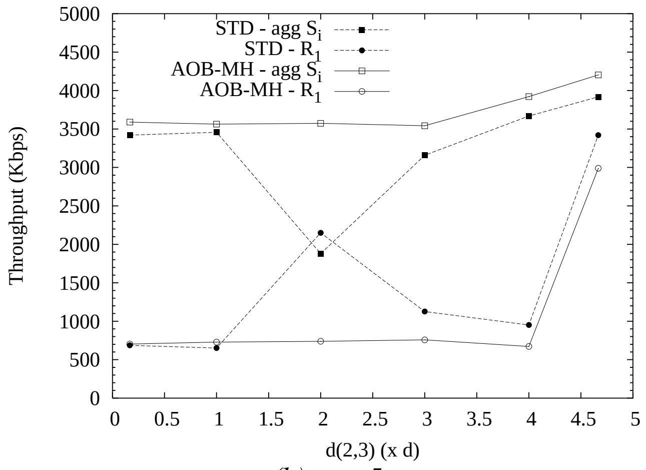
$S\_U_{Int}^{k}$ value and the higher could be the potential unfairness in the network. As a consequence, stations that achieve large $S\_U_{Int}^{k}$ should be less aggressive in transmitting than stations with small $S\_U_{Int}^{k}$, such that these stations have a higher probability to access the channel and use their credits. Specifically, according to formula (14), if all the transmitters obtain the same $S\_U_{Int}^{k}$, then they would release transmission attempts with the same probability. On the other hand, if one of the transmitters has higher internal slot utilization than the other transmitters because the standard MAC protocol fails in providing a fair coordination of the channel access among the stations, then that transmitter would release on average more transmission attempts than the other transmitters. As a consequence, the advantaged stations release channel time for allowing additional channel accesses to the disadvantaged stations in the network. At this stage of our study we have assumed that all the stations are cooperative, hence all of them behaves following formula (14).
It is worth pointing out that the P_T as defied in formula (14), hinders the stations to reach the optimal slot-utilization level identified by the ACL value. However, the credits scheme contributes to balance the increase in the probability of releasing transmission opportunities with respect to the original AOB mechanism.

2.10.1. Performance Evaluation

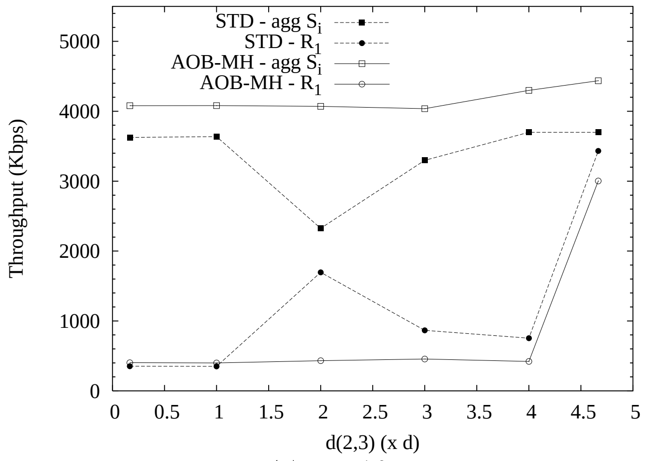
In this section, by means of discrete event simulations, we extensively investigate the performance of the extended AOB mechanism; henceforth indicated as AOB-MH, comparing it with the standard MAC protocol. Table 1 lists the parameters' setting used during the simulations. As far as the AOB-MH-specific parameters, the stations are allowed to transmit at most five consecutive immediate frames (i.e., $l = 5$), if enough credits are available. The transmission and carrier sense ranges we used are the same considered in Section 2.8.1. If not otherwise specified, the packet size is constant and equal to 576 bytes.
In the following figures, we show the aggregate throughput obtained by stations S_i, with $i \in \{1,\ldots,n\}$, comparing it with the throughput achieved by the station R_1. All the curves referring to the standard MAC protocol are labeled as *STD*, while all the curves referring to the AOB-MH mechanism are labeled as *AOB-MH*. Performance figures have been estimated with the independent replication technique with a 95 percent confidence level. Confidence intervals are not reported into the graphs, as they are always very tight (≤ 1 percent).



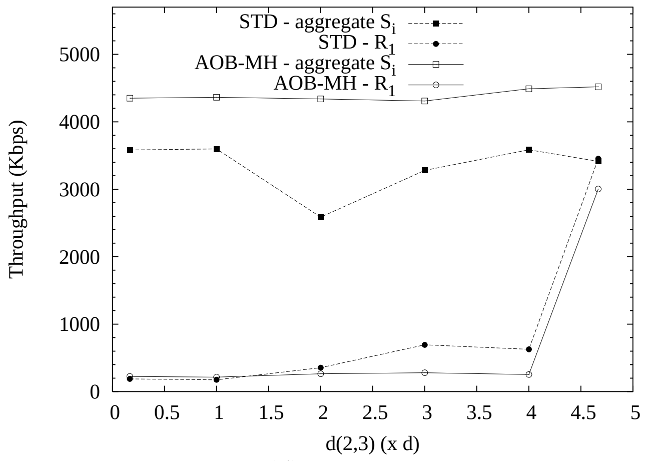
(a) $n = 1$.



(b) $n = 5$.



(c) $n = 10$.



(d) $n = 20$.

Figure 2.20: Sessions' throughputs as functions of the distance $d(2,3)$ *for the network scenario shown in Figure 2.19, comparing the* extended *AOB-MH mechanism and the standard protocol.*

Figure 2.20(a) compares the throughputs obtained by stations S_1 and R_1, when $n = 1$, that is the network scenario also depicted in Figure 2.4. From the shown results, we can observe that the AOB-MH mechanism is able to enforce a fair channel access between the two transmitters up to $d(2,3) = 3d$. However, for $d(2,3) = 4d$ the unfair channel allocation still exists. This can be explained by observing in Figure 2.17(a) and Figure 2.17(b) the slot utilization in the case of $d(2,3) = 4d$ (i.e., $d(2,3) = 120$ m). In particular, since S_1 is outside the carrier sensing range of R_1, S_1 observes larger idle periods on the channel than R_1, the overall slot utilization S_U computed in S_1 is lower than the one computed in R_1. As a consequence, the P_T used by S_1 is greater than the one used by R_1, resulting in a higher throughput.[12] Finally, we note that the fairness in the network has been obtained at the cost of a slightly decrease in the total channel utilization.

Figure 2.20(b), Figure 2.20(c) and Figure 2.20(d) compare the aggregate throughput obtained by stations S_i, with $i \in \{1,\dots,n\}$, and the throughput achieved by station R_1, when $n = 5$, $n = 10$ and $n = 20$, respectively. This experiments are finalized to verify that the AOB-MH mechanism effectively ensure a fair share of the channel among the stations in the network, independently of the number of senders that are exposed to R_1's transmissions. The curves clearly indicate that the AOB-MH significantly improves the performance of the standard MAC protocol because: *i)* the throughput of stations S_i and R_1 are almost independent of the distance $d(2,3)$, with a fair share of the channel bandwidth among the transmitters; *ii)* the deep unfairness observed at the distance $d(2,3) = 2d$, caused by the channel asymmetry due to the large $EIFS$ timer, is completely solved; *iii)* the contention control performed by releasing transmission opportunities that could increase the channel utilization beyond the optimal slot-utilization level identified by the ACL value, is effective in increasing the network capacity.

2.11. Enhanced card novel architecture & mechanisms

During the first project year (see deliverable D5), 3 main tasks have been performed in order to provide a new medium access technology (experimental verification with laboratory and field tests) to MobileMAN, namely:

- State-of-the-art investigation
- Choice of a hardware medium-access platform for MobileMAN
- Preliminary decisions for a flexible architecture for the packets management

During the second project year, the actual implementation has been carried out accordingly. More in details, work has been performed in 2 fields:

- Medium access platform
- Packet management architecture

2.12. Medium Access platform

The hardware specified at the end of the first project year is a combination of a modified version of the DT-20 wireless modem (made and customized by Elektrobit AG) with a compact DSP board (a standard Orsys GmbH product). Figure 2.21 shows a simple setup with 2 nodes.

12 In this case, as the stations operate in channel areas corresponding to different contention levels this difference thus not implies any unfair behavior of the MAC. It can be expected that a more uniform view of the channel can be achieved by exploiting cross layer interactions among the MAC and the network layer. For example the routing protocol may distribute in addition to existing links also the slot utilization level corresponding to that link.

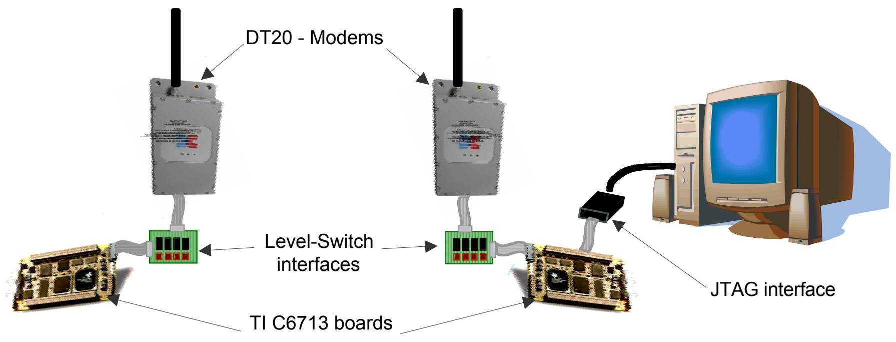


Figure 2.21. Medium Access platform (2 items connected by coaxial cable for lab tests).

The host computers running the user applications (not shown in Figure 2.21) should be further connected to the DSP board through the IEEE1394a (FireWire®) port. The software development system is connected to the DSP boards through the JTAG port; this is necessary during the firmware development phase, when the firmware (communication and MAC software) is build and downloaded into the on-board non-volatile (FLASH) memory.

Since the DSP board includes a C6000 platform floating-point processor, the Texas Instruments *Code Composer Studio* has been selected for the firmware development.

The firmware running on the C6713 DSP is basically a simple monitor loop with few interrupt routines. This allows a more efficient exploitation of the processor resources. The nature of the MAC software is such that an OS is not necessary; however, a Real-Time Operating System (RT-OS) could be added in future if necessary.

A total of 4 systems have been assembled until now.

Completed work

During the second project year, the following work has been completed on the Medium Access Control platform:

- Deep analysis of the 802.11 standard
- Flowcharts and procedures defining the Tx/Rx
- Implementation of the monitor loop
- Implementation of the 802.11 MAC CRC on the embedded FPGA (this is an Intellectual Property block with source written in VHDL)
- Implementation of Tx routines (MAC to BB/RF modem)
- Implementation of Rx routines (BB/RF modem to MAC)
- Implementation of the channel sensing mechanism (signals from RF part)
- Implementation of the standard 802.11 backoff mechanism
- Implementation of the regular 802.11 frame generation
- Implementation of the fragmented frame generation
- Implementation of the RTS/CTS/DATA/AK handshake
- Implementation of the MAC Address recognition
- Implementation of the channel contention mechanism

The realized firmware gave the possibility to perform a first series of tests on the actual systems: the correct firmware functionality has been successfully verified with the help of following tests (2 platforms where connected through a *coaxial cable* in a "laboratory set-up"):

- Stress test of Tx between 2 systems
- Stress test of Rx between 2 systems
- Stress test of alternating Tx/Rx between 2 systems

Moreover, tests have been performed in wireless mode also, with the 2 systems connected through off-the-shelf WLAN antennas; following functionalities have been successfully verified:

- Alternating Tx/Rx between 2 systems
- Tx/Rx with standard backoff mechanism activated (collisions were artificially forced)
- Fragmented Tx/Rx transmissions
- Handshake mechanism in RTS/CTS/DATA/AK transmissions

Current work

Currently a phase of specification is defining the required components in order to use the systems in ad-hoc mode and with real (test) applications running on the host computer. This includes:

- Definition of the host interface mechanism (the host is connected to the C6713 DSP via IEEE1394a).
- Definition of a simple packet data-structure for basic functionality. This will allow the management of packets in both the standard 802.11 and the modified (CNR Pisa) proposed MAC layers; conventional Tx and Rx packets will be managed this way.
- Definition of data-structure extensions for cross-layering implementation. This will allow to later add extra functionalities at the MAC layer level (e.g. Filtering, part of the routing, packet priority queues, Quality of Service...) witch are not defined in the IEEE802.11 standard, but that could be very useful (and used) for MobileMAN.

Next steps

The next steps concerning the development of the Media Access platform include:

- Implementation of the NAV (Network Allocation Vector)
- Test of a 3 nodes network with above functionality in wireless mode
- Implementation and test of the host interface mechanism
- Implementation and test of the standard packets data-structure
- Implementation and test of the modified (CNR) backoff mechanism
- Implementation and test of data-structure extensions for cross-layering

The first 5 steps are particularly important for the critical evaluation of MobileMAN. In facts, one of the main advantages of MobileMAN with respect to the IEEE802.11 standard is the much better performance in terms of bandwidth utilization (as shown by the CNR-Pisa simulations). The main responsible for such dramatic improvement is the modified backoff mechanism of the MAC layer. In order to verify this hypothesis, comparative performance measures of the MobileMAN MAC layer and of standard IEEE802.11 MAC layer should be done, and for this the MAC firmware must be completed (the first 5 steps in the above list). It should be noted that the full implementation of the standard IEEE802.11 MAC is necessary in order to have common working conditions for both MAC layers: an off-the-shelf WLAN card would have too many different parameters, yielding useless measures.

2.13. Packet management architecture

For the software, a flexible architecture is under development, i.e. an architecture which not only allows the implementation of the MAC software alone, but which also allows flexibility and a future extensions. Figure 2.22 shows simplified schematics of the MAC card software architecture.

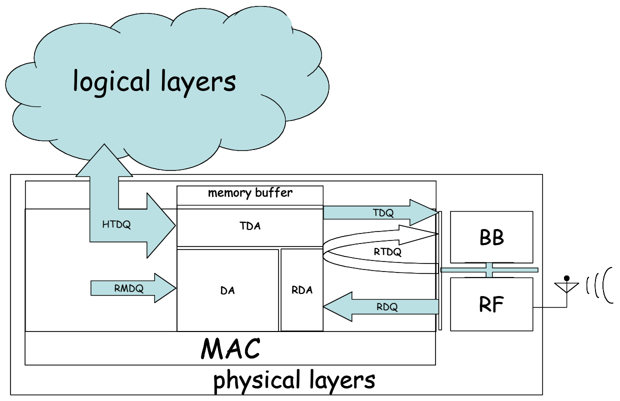


Figure 2.22. MobileMAN (enhanced) MAC card software architecture (simplified view).

The processor unit (PU), which supports this new MAC, has to manage the data flow between the communication channel and the host (logical layers) in both directions and channel-to-channel in case of atomic operation (e.g. RTS/CTS). This means that the PU needs a specialized data structure. Since the communication must also satisfy the stringent time constraints and frames hierarchy imposed by the IEEE802.11 standard, the data structure must be optimized in terms of numbers of memory access, numbers of data swaps and storage mechanism.

It has been chosen to implement a buffer management scheme with a descriptor mechanism; this allows to efficiently process receive- and transmit data packets in place, and to eliminate packet copy or swapping. The buffer memory is configured in three different areas:

- Data area (DA) for storing data from the logical layers to the PHY, frames ready to be transmitted, frames received from the channel and data from the PHY to the logical layers.
- Transmission descriptor area (TDA) for storing descriptors pointing to the data packets ready for transmission.
- Receive descriptor area (RDA) for storing all descriptors pointing to each received data packet.

Each descriptor points to a specific data packet/frame and all descriptors are arranged in different queues: transmission descriptor queue (TDQ) for the transmission on the channel, host transmission descriptor queue (HTDQ) for the transmission to the logical layers, receive descriptor queue (RDQ) for the received frames, receive transmission descriptor queue (RTDQ) for those frames of an atomic operation, and reusable memory descriptor queue (RMDQ) for the reusable data area. In this way, the PU manages priorities, hierarchy and sequence of the frames/data packets.

Thanks to the descriptor mechanism, the PU doesn't need to move, swap, copy or erase data in memory, but it has only to change few flags in the descriptor's control and status word or to change the pointers (address registers) which are also part of the descriptor. The defined data structure is easily extendable in order to accommodate new MAC features (currently under investigation by other MobileMAN project partners from a theoretical point of view), as for instance:

- Cross layering: several mechanisms can profit by the knowledge of some parameters that are typically confined at the MAC layer, like transport, power management, cooperation, etc.
- MAC-level routing: a packet received at the wireless interface must be passed up to the routing layer (in order to discover the next hop), and further down to the same wireless interface for transferring it to the next hop; this adds undesirable delay and overhead at both MAC and routing layer.

Current work

Currently, the data structure and the associated management mechanism are in the phase of refinement. A first specification of both the data structure and the management software will be produced at the beginning of the third project year.

Next steps

Once the first specification will be available, the corresponding software will be written (data structure definition and management routines) and integrated into the firmware of the realized Media Access platform.

2.14. MAC development platform hardware and software description

2.14.1. Hardware

As stated above, the hardware is a combination of a modified version of the DT-20 wireless modem (made and customized by Elektrobit AG) with a compact DSP board (a standard Orsys GmbH product). Figure 2.23 shows 2 platforms connected by a coaxial cable for test and verification purposes.

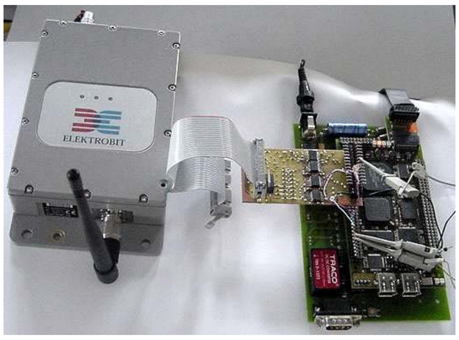


Figure 2.23. Medium Access platform. RF modem and DSP board are connected through a small custom-made adapter board (level shifters for voltage level conversion.

The DT-20 modem performs the base-band and the RF processing thanks to an Intersil Prism-I WLAN chipset.

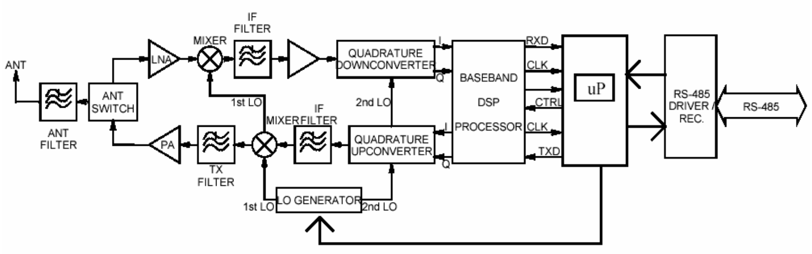


Figure 2.24. Block diagram of the RF modem (customized Elektrobit AG DT-20 modem).

Actually, in the RF modem, there is an on-board DSP microcontroller (which is used for a very simple special communication protocol); this DSP is however not used in MobileMAN since it is not enough powerful to accommodate the required IEEE802.11 firmware (plus the extensions brought by MobileMAN). The modem was customized by Elektrobit; the baseband input and output signals (synchronous serial lines) where brought to the external world and connected through flat-cables to the external DSP microprocessor board, where the actual MAC firmware is implemented.

The DSP board (Orsys GmbH) is based on the TIC6713 DSP microprocessor (Texas Instruments).

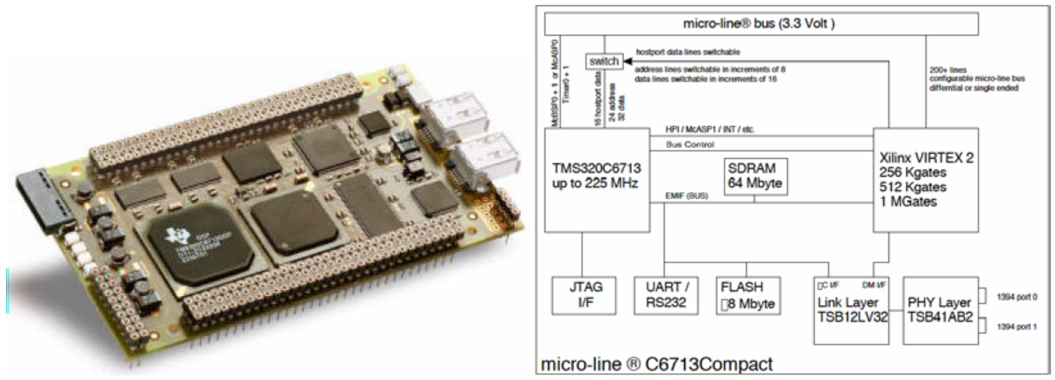


Figure 2.25. RF modem. The customized Elektrobit AG DT-20 modem is shown with its block schematics.

The board includes, among other, a FLASH memory for the non-volatile storage of the firmware, an extension RAM and a large FPGA. This last component has been used for the acceleration of the CRC checksum computation, and could be used in the future in order to accelerate other tasks (e.g. address filtering, cryptography...)

Physically, the DSP board is connected to the RF modem through a synchronous serial interface (a standard SPI); a small custom-made level shifter board is necessary in order to adapt the logic signals voltage levels (3.3V for the DSP board and 5V for the Modem). The choice of the SPI as communication channel between DSP and RF-modem is mandatory, since the Intersil Prism-I chipset has only this interface. However, the DSP board has much more fast communication channels (I/O ports, DMA...), which could be used for future developments (in case a different and faster RF hardware, such the IEEE802.11b or the IEEE802.11g, would be chosen).

The communication with the host computer is done through an IEEE1394a (FireWire®) port which is capable to reach a transfer bitrate to 400 Mbps (10^6 bit/s), namely 200 Mbps in each direction; this is largely sufficient for the MobileMAN project, and is certainly sufficient even for future extensions, where a faster BB/RF front-end (e.g. IEEE802.11b) could be used. The FireWire® channel is used as a simple pint-to-point channel between host CPU and Medium Access platform: the packets are just tunnelled through the FireWire® link. The host CPU and the Medium Access platform are seamlessly integrated: the customization of the upper protocols layers (e.g. TCP/IP) on the host CPU is trivial.

2.14.2. Software

Actual firmware

Actually, the DSP board performs 3 tasks:

1) MAC protocol implementation.
2) Communication with the wireless modem.
3) Communication with the host CPU.

The main and most critical task is clearly the MAC protocol implementation; several tasks have to be carried out in real-time in order to respect the strict IEEE802.11 standard. As stated before, the software is actually based on a simple monitor with interrupt routines; no Operation System is used, an RT-OS could however be easily added is necessary.

The communication with the wireless modem is basically done by 2 interrupt routines (one for Tx and one for Rx). The Tx routine ensures that the BB input FIFO does not underflow while transmitting a packet (or a fragment); this is easily achieved with the current C6713 DSP for the 2 Mbps IEEE802.11 standard: a 32 bit word should be written into the SPI transmit FIFO every 16 µs. Accordingly, a 32 bit word should be read out of the SPI receive FIFO every 16 µs. The C6713 DSP could easily accommodate more demanding standards such the 11 Mbps IEEE802.11b.

The communication with the host CPU is done via the on board IEEE1394a port. With this 400 Mpbs (200 Mbps in each direction), there is a large margin for the DSP; in facts, the DSP writes and reads directly to the IEEE1394a chipset registers.

Packet management

The packet management software is currently under development and is not actually part of the Firmware. The memory is configured in three different areas: data area (DA), transmission descriptor area, (TDA), and receive descriptor area, (RDA).

A) Data Area

The Data Area stores all the packets/frames that are flowing in all direction: from the host to the channel, from the channel to the host and from the channel to the channel. The latest case concern the atomic operation like RTS/CTS and AKN and there must to be managed immediately. For this reason there is a small amount of memory at the end of the DA, Atomic Data Area (ADA), where these frames are temporarily stored.

All the other packets/frames are stored in the Normal Data Area, (NDA), in the same order as they arrive. The storing mechanism followed by the PU is explained below:

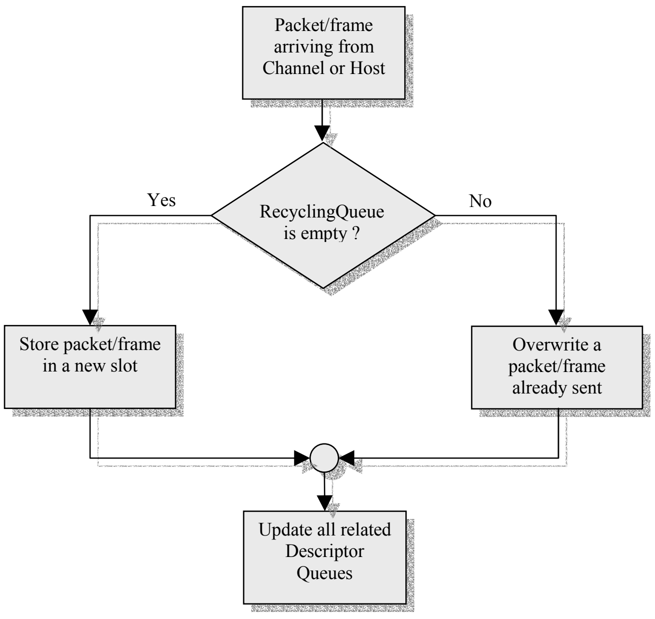

Packet/frame arriving from Channel or Host
RecyclingQueue is empty ?
Yes
No
Store packet/frame in a new slot
Overwrite a packet/frame already sent
Update all related Descriptor Queues

Figure 2.26. Storing mechanism flowchart.

The length of DA is parameterized with respect to the user needs and the total memory size, and its value is stored in *MaxDALength* variable. The NDA is slotted and the length of each slot is defined by the *MaxFrameLength* variable that is defined by the user. In this way the same structure could be reused for different hardware implementations and different standards.

The NDA is also split up in two areas with different length: the bigger is the first 80% of the total NDA length and the smaller is the last 20%. The smaller area is called Guard Normal Data Area, (GNDA), and is used when the first 80% of the NDA is full. When that happens the PU knows that the memory is closed to the saturation, so it could stop the incoming data from the host until all the data prepared for the transmission on the channel has been sent.

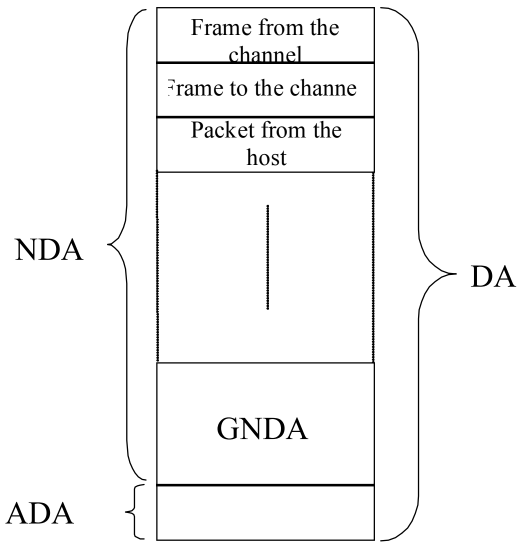


Figure 2.27. Data Area structure.

Data storage

The NDA slots have all fixed length, which is the maximum frame length, but the data coming in the memory have variable length: variable MAC header and variable payload. Moreover, the packets coming from the host are only the payload because the MAC header and the FCS are inserted later by the PU, above and below the payload respectively. Since the FCS is the only part that has always the same length, four bytes, the addresses for the first byte of the payload and the first byte of the MAC header are determined as follow:

$$A_{payload} = A_{start\_slot} - (A_{end\_slot} - (4 + N_{byte\_payload})),$$

$$A_{MAC\_h} = A_{payload} - N_{byte\_MAC\_h}$$

where

$A_{payload}$	address of the first byte of the payload
A_{start_slot}	address of the beginning of the memory slot, (stored in the descriptor)
A_{end_slot}	address of the end of the memory slot, (stored in the descriptor)
$N_{byte\_payload}$	number of bytes of the payload, (stored in the descriptor)
A_{MAC_h}	address of the first byte of the MAC header
$N_{byte\_MAC\_h}$	number of bytes of the MAC header

B) Transmission and Reception Descriptor Areas (TRDA)

The transmission and reception descriptor areas are differentiated from a logical point of view, but the descriptors are stored in the same room.

All the descriptors are arranged in three different queues: transmission descriptor queue (TDQ), reception descriptor queue (RDQ) and empty descriptor queue (EDQ). Furthermore, TDQ and RDQ are organized as queue of queues where each queue is related to a certain level of priority that is previously assigned to the packet/frame. At the moment there are 7 levels of priority available, in which 0 is the higher level.

The organization of TDQ and RDQ is illustrated in the Figure below.

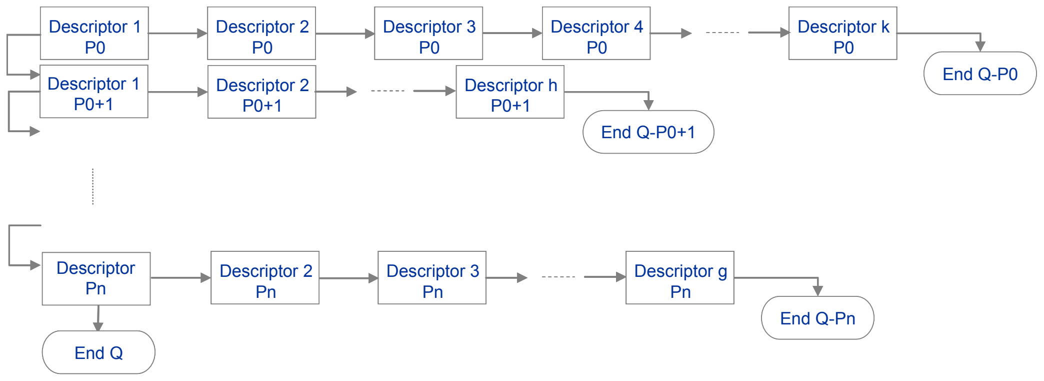


Figure 2.28. TDQ and RDQ structure; P is the priority level.

At the beginning of the TRDA there are 60 bytes assigned to 15 pointers, 4 bytes each one, that have the following meaning:

- 1 to 7: pointers to the beginning of the TDQs depending on their priority level.
- 8 to 14: pointers to the beginning of the RDQs depending on their priority level.
- 15: pointer to the beginning of the EDQ.

The descriptors have all the same dimension, which is 23 bytes, and they are organized as follow:

Register name	**N° Bytes**	**Description**
Control	2	Control flags
Address_start_slot_H	2	High part address of the beginning of the memory slot pointed by the descriptor
Address_start_slot_L	2	Low part address of the beginning of the memory slot pointed by the descriptor
N_byte_payload	2	Number of bytes of the payload
Address_payload	2	Address of the payload in the slot
N_byte_MAC_heder	1	Number of bytes of the MAC header
Address_MAC_header	2	Address of the MAC header in the slot
Reserved	6	Reserved for future implementation
Address_next_descriptor_H	2	High part of the address of the new descriptor in the queue
Address_next_descriptor_L	2	Low part of the address of the new descriptor in the queue

2.15. References

[ABCG04] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "Wi-Fi in Ad Hoc Mode: A Measurement Study", in *Proceedings of PerCom 2004*, Orlando, FL, March 14-17 2004, pp. 145-154.

[B00] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function", *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 9, 2000, pp. 1787-1800.

[BCD00] L. Bononi, M. Conti, L. Donatiello, "Design and Performance Evaluation of a Distributed Contention Control (DCC) Mechanism for IEEE 802.11 Wireless Local Area Networks", *Journal of Parallel and Distributed Computing*, vol. 60, no. 4, pp. 407-430, April 2000.

[BCG02] R. Bruno, M. Conti, E. Gregori, "Optimization of Efficiency and Energy Consumption in p-Persistent CSMA-Based Wireless LANs", *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, pp. 10-31, March 2002

[BCG03] R. Bruno, M. Conti, E. Gregori, "Optimal Capacity of *p*-Persistent CSMA Protocols. *IEEE Communication Letters"*, vol. 7, no. 3, March 2003, pp. 139-141.

[BCG04] L. Bononi, M. Conti, E. Gregori, "Run-Time Optimization of IEEE 802.11 Wireless LANs Performance", *IEEE Transactions on Parallel and Distributed Systems*, vol 15, no. 1, pp. 66-80, 2004.

[BH03] L. Buttyan, J, Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks", *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 8, no. 9, pp. 1774-1786, 2000.

[BWK00] R. Bensaouu, Y. Wang, C.-C. Ko, "Fair Medium Access in 802.11 based Wireless Ad-Hoc Networks", in *Proceedings of IEEE Mobihoc'00*, Boston, MA, August 11 2000, pp. 99-106.

[CCG00] F. Calí, M. Conti, E. Gregori, "Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit", *IEEE/ACM Transactions on Networking*, vol. 8, no. 6, December 2000, pp. 785-799.

[CCG00a] F. Calí, M. Conti, E. Gregori, "Dynamic IEEE 802.11: design, modeling and performance evaluation", *Journal on Selected Areas in Communications*, vol. 18, no. 9, September 2000, pp. 1774-1786.

[CGKO04] J. Crowcroft, R. Gibbens, F. Kelly, S. Östring, "Modeling incentives for collaboration in mobile ad hoc networks", *Performance Evaluation*, vol 54, no. 4, pp. 427-439, August 2004.

[D01] E. Dutkievicz, "Impact of Transmit Range on Throughput Performance in Mobile Ad Hoc Networks", in *Proceedings of ICC 2001*, vol. 9, Los Angeles, LA, June 11-14 2001, pp. 2933-2937.

[FZLZG03] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wirelss Channel on TCP Throughput and Loss", in *Proceedings of IEEE Infocom 2003*, vol. 3, San Francisco, CA, March 30ñ April 3 2003, pp. 1744-1753.

[IEEE99] ANSI/IEEE, Piscataway, NJ, Std. 802.11, "*Wireless LAN Medium Access Control and Physical Layer (PHY) Specification*", 1999.

[IEEEb01] ANSI/IEEE, Piscataway, NJ, Std. 802.11*b*, "*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification/ Amendment 2: Higher-speed Physical Layer (PHY) in the 2:4 GHz band*", 2001.

[Ns2] The Network Simulator (*ns-2*), version 2.27 - VINT Project - http://www.isi.edu/nsnam/ns/index.html.

[QN03] Qualnet Simulator, version 3.7 - Scalable Network Technologies Inc. – http://www.qualnet.com/.

[WJCD00] C. Ware, J. Judge, J. Chicaro, E. Dutkievicz, "Unfairness and Capture Behaviour in 802.11 Ad Hoc Networks", in *Proceedings of ICC 2000*, vol. 1, New Orleans, LA, June 18-22 2000, pp. 159-163

[WWC01] C. Ware, T. Wysocki, J. Chicaro, "Hidden Terminal Problems in IEEE 802.11 Mobile Ad Hoc Networks", in *Proceedings of ICC 2001*, vol. 1, Los Angeles, LA, June 11-14 2001, pp. 261-265.

[XGB02] K. Xu, M. Gerla, S. Bae, "How Effective is the IEEE 802.11 RTS/CTS Handshake in Ad Hoc Networks" in *Proceedings of GLOBECOM 2002*, vol. 1, Taipei, Taiwan, November 17-21 2002, pp. 72-76.

[XGB03] K. Xu, M. Gerla, S. Bae, "Effectiveness of RTS/CTS handshake in IEEE 802.11 based ad hoc networks", *Ad Hoc Networks*, vol. 1, no. 1, pp. 107-123, July 2003.

[XS01] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol Work Well in Multihop Wireless Ad Hoc Networks?", *IEEE Communications Magazine.*, vol. 39, no. 6, pp. 130-137, June 2001.

[XS02] S. Xu and T. Saadawi, "Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks", *Computer Networks*, vol. 38, no. 1, pp. 531-548, July 2002.

[ZBG98] X. Zeng, R. Bagrodia, M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks", in *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS'98)*, Banff, Canada, May 26-29 1998, pp. 154-161.

[ZCY03] S. Zhong, J, Chen, Y. Yang, "SPRITE: A Simple, cheat-Proof, Credit-Based System for Mobile Ad Hoc Networks", In *Proceedings of IEEE Infocom 2003*, San Francisco, CA, March 30-April 3 2003, pp. 1987-1997.

3. NETWORKING

In a MANET, to cope with the self-organizing, dynamic, volatile, peer-to-peer communication environment, most of the main functionalities of the *Networking protocols* (i.e., network and transport protocols in the Internet architecture) need to be re-designed.

The aim of the networking protocols is to use the one-hop transmission services provided by the network-interface-card technology to construct end-to-end (reliable) delivery services, from a sender to the receiver. To establish an end-to-end communication, the sender needs to locate the receiver inside the network. The purpose of a *location service* is to dynamically map the logical address of the (receiver) device to its current location in the network. Once, a user is located, *routing and forwarding algorithms* must be provided to route the information through the MANET. Finally, the low reliability of communications (due to wireless communications, users' mobility, etc.), and the possibility of network congestion require a *re-design of Transport Layer mechanisms.*

During the first year of the project (see D5) we identified existing protocols that can be exploited to achieve the project targets and we designed our own solutions whenever necessary. Specifically, we identified two legacy routing protocols, AODV and OLSR, and provided the preliminary design of a reliable forwarding scheme (named REEF) and a new transport layer protocol (named TPA). The work during the second year focused on enhancing the Ad Hoc framework architecture to support/exploit cross layer interactions. Specifically, the main contributions of this activity have been:

- The extension of the framework with a proactive routing protocol based on *Limited Dissemination* policies, named **Hazy Sighted Link State (HSLS)**, in which routing updates are flooded in the network with a binary exponential sequence.
- The extension of link-state mechanism (used by proactive routing protocols) to make LSU packets able to carry *optional information* as piggybacked data, such as services' information. By exploiting this LSU extension in our cross-layer architecture we designed an effective Service Discovery protocol.
- The definition of an efficient mechanism, named REEF, for the reliable forwarding of data on the (possible) multi routes between a sender and a receiver. REEF performs its task by exploiting the information produced by the routing protocol and cross-layer interactions with the transport protocol.

During the third year we completed the design and validation of the TPA protocol (see Section 3.4). Specifically, TPA validation was performed via simulation by comparing its performance with those of a legacy TCP protocol. Results indicate that in all scenarios we considered TPA outperforms legacy TCP.

Hereafter, we present and discuss the solutions that we have designed and implemented in the framework of the MobileMAN project. A description of legacy protocols that we used in the MobileMAN architecture can be found in D5 and [28].

3.1. Routing in a cross layering architecture: HSLS

The research community has developed a set of routing protocols specifically for Ad Hoc networks. In a rough classification, Ad Hoc routing protocols can be classified in *proactive*, *reactive* (also called *on-demand*) and *hybrid* approaches. In the proactive strategy the route state information is periodically exchanged among hosts (e.g. DSDV, OSLR), allowing each node to build a global knowledge of the network independently of the actually used routes. On the other hand, on-demand approaches limit the exchange of route state information, building routes only towards nodes involved in higher layers communication (e.g. AODV, DSR) [28].

As discussed in [1], if we focus on the routing function in isolation on-demand protocols are preferred as they provide much better scalability when the network topology is extremely dynamic and nodes do not need to store link state information. On the other hand, proactive protocols do not scale with large networks (due to the significant amount of information needed to collect global routing decisions) but through periodic dissemination of LSU packets provide to a node a more complete knowledge of network topology; furthermore delays for information transfer may be significantly shorter with respect to on-demand protocols.

In cross-layer architecture, the richer amount of information collected by proactive protocols can be exploited, for purposes other than routing, at other layers. Overheads cannot be evaluated in isolation focusing on a certain level, but new cross-layer metrics must be applied. An example of this concept could be the service location for middleware: once the routing protocol has discovered the topology of the network, the middleware can use it to identify the node that provides a certain service without performing a new route discovery. We can identify many other examples that clearly indicate the advantages for a node to have knowledge of the network topology. The result is that proactive approaches may better satisfy the self-organizing requirement of general-purpose Ad Hoc networks.

In the framework of the cross-layer MobileMAN architecture, we worked to identify, if possible, a routing protocol suitable for multi-hop networks in terms of scalability, performance and efficiency, but also able to provide a rich set of information about the network that can be exploited to improve the other protocols of the cross-layer architecture.

To cope with scalability problems we investigated a class of proactive routing protocols based on a link-state dissemination policy. Generally, a link-state routing algorithm consists in each node broadcasting packets containing an up-to-date version of its one hop neighborhood configuration. As these packets, called link-state updates (LSU), are flooded throughout the network, each node is able to synthesize the overall topology. The dissemination of LSUs (i.e. continuous broadcast) may cause a scalability issue, but a significant overhead's reduction can be achieved by controlling the *scope* and the *frequency* of floods. Intuitively, LSUs are flooded with increasing Time-To-Live (TTL) for decreasing frequencies. In such a way, neighbor nodes receive LSUs more often respect to far away ones. This is motivated by the fact that in hop-by-hop routing, far away topology changes have little impact in a local node next hop decision. As a result, each node builds a "self-centered" topology view, which becomes *hazy* as the distance grows. Recent studies ([5], [6]) have analytically proved that the family of *Link State* routing protocols based on *Limited Dissemination* exhibit good performance in term of scalability. Examples of this approach are hierarchically link state [4], FSR [3], GSR [2]. In particular, as proved in [5], the best among them is the **Hazy Sighted Link State (HSLS)** [6] in which routing updates are flooded in the network with a binary exponential sequence.

3.1.1. HSLS Overview

In HSLS, similarly to others proactive protocols, each node sends periodic route updates (LSU packets) containing its one-hop neighbourhood, allowing other nodes to have a complete view of the network; but with the aim to reduce the overall control overhead, and have good scalability properties, there is a restriction of the scope of routing updates in time and/or space. Specifically, periodically each node broadcasts the list of its 1-hop neighbours over the network with a frequency that decreases with distance. Thus each node has a partial knowledge of the topology (i.e. not real-time uploaded); it is more precise in the nearby and more hazy far from a node; this strategy, if coupled with a forwarding strategy that in each node independently selects the next hop towards a destination (see, for example, the reliable forwarding strategy presented in Section 3.3), is expected not causing any major impact on the selection of the path towards the destination.

In HSLS periodic updates occur at discrete time interval. A node collects one or more link status changes in a single packet which is transmitted only at particularly time instants that are multiple of t_e seconds. Furthermore, the dissemination of this information is controlled by specifying the area of the network in which the Link State Update (LSU) will be distributed. This control is implemented by setting the TTL *(Time To Live)* field of the LSU packets thus limiting the number of hops the packet will perform in the network. More precisely, let us indicate with 0 the time instant at which a node sends a *global LSU* (packet that travels over the entire network), providing a complete knowledge of link changes to all nodes in the network, then a node wakes up:

- every t_e seconds, and transmits an LSU with field TTL *(Time To Live)* equal to 2 if there has been a link status change in the last t_e seconds;
- every 2 t_e seconds, and transmits an LSU with field TTL *(Time To Live)* equal to 4 if there has been a link status change in the last 2 t_e seconds;
- ….

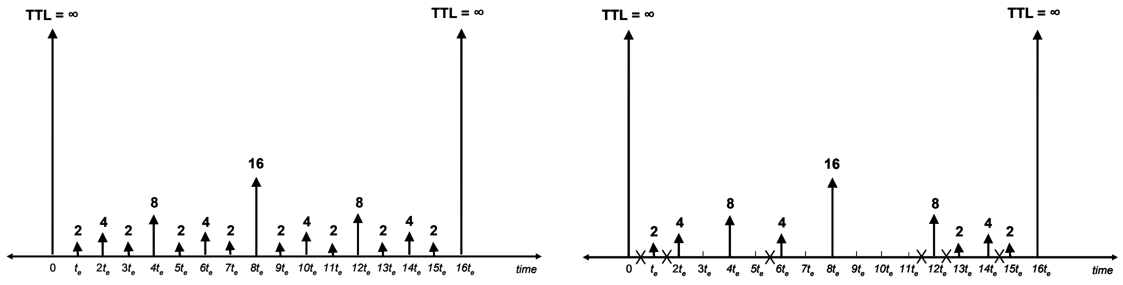


Figure 3.1: Behavior of the HSLS algorithm in different scenarios

In general, a node wakes up every 2^i t_e seconds (with i = 0, 1, 2, 3) and sends an LSU with TTL = 2^{i+1} if there has been a link status change in the last 2^i t_e seconds. If the value 2^{i+1} is greater than the distance from this node to any other node in the network, the TTL field is set to infinity (i.e., a *global LSU*) and all counters and timers are reset.

Figure 3.1 shows some examples of HSLS's LSU generation process. In the left figure we assume a high mobility scenario in which a link change occurs every t_e seconds, and hence LSU packets, represented by vertical arrows, are sent in the network every t_e seconds; The height of the arrow represents the TTL value.
On the other hand, the right hand side of Figure 3.1 represents a lower mobility scenario in which there is not a link change every t_e seconds. Specifically, changes are marked with an 'x' on time axis and, as it appears in the figure, that LSU packets are less frequent, and it may happen that some updating points are skipped if in the last interval not change occurred

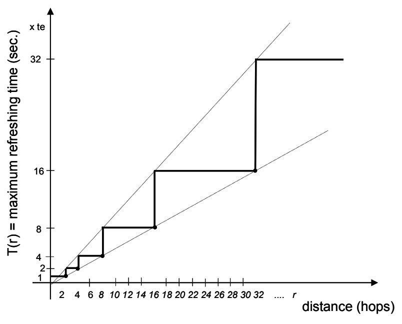


Figure 3.2: Maximum refresh time as a function of distance from link event

The above approach guarantees that 2^{i+1} hops neighbours from a tagged node will realized topology changes at most after $2^i t_e$ seconds. Figure 3.2 shows the latency in propagation of link state information performed by HSLS protocol.

3.1.2. Reliable HSLS

Generally, link state routing protocols don't provide any form of acknowledgement for the control packets because link state information is spread in the network using a broadcast process. Moreover, the 802.11 MAC protocol delivers broadcast packets in an unreliable fashion, i.e. without an explicit acknowledgement. Therefore if a node sends an LSU and it is lost due to collisions or channel interference, that packet is never retransmitted neither at network, nor at link layer. To guarantee a reliable delivery of LSU packets, a reliability mechanism to should be added to HSLS. Hereafter, we present our approach to improve HSLS reliability in an efficient way.

Instead of introducing additional control packets, broadcasted LSUs are used as acknowledgements of the LSU previously sent. In order to record the history about LSUs sent and received from the network, each node stores them in two caches: *sentLSUcache* maintains information about LSUs generated by the node itself; instead *receivedLSUcache* stores LSUs coming from other nodes.

Referring to Figure 3.3, let's suppose that a node *X* is a originator of an LSU packet; after its reception, a generic node A forwards it with $TTL = i$. Node A will consider an ACK for this LSU packet, *ACK_LSU*, any LSU packet it will receive from its 1-hop neighbours with originator node X and $TTL = i-1$.

More precisely, the following procedure is executed on each node to guarantee reliability of the LSU dissemination process:

1. Node A sends an LSU with $TTL = i$;
2. Node A counts the number of ACK_LSU packets received from its 1-hop neighbours during a fixed time window $T \ll t_e$; in particular, it stores the number of received ACK_LSU for each LSU into *sentLSUcache* if it is the originator of this LSU, or into *receivedLSUcache* otherwise;
3. If the number of ACK_LSU >= ACK_threshold, it can be assumed that the LSU sent by A has been correctly received from most of its neighbours; on the other hand, if the number of ACK_LSU < ACK_threshold node A has to retransmit the same LSU again;
4. An explicit ACK_LSU is sent in the last hop.

The fourth point assures the uniformity in the reliability process also in the last hop. Suppose that node C (in Figure 3.3) forwards an LSU packet with $TTL = 0$; nodes D and E receive and process it, but they will not forward it anymore because the *TTL* value. Consequently, the timeout *T* at node C would expire without having received any acknowledgment. Thus, node C would make a wrong decision to retransmit that LSU packet. To avoid this, node D and E will send an explicit ACK_LSU (a copy of the received LSU without message body, named ACK_expl) to the sender C (i.e., unicast transmission) as confirmation of their previously correct reception. In this way only negligible additional control traffic is added to the original protocol.

The ACK_threshold should be a value between 1 and the number of 1-hop neighbours. For instance, using the lowest value 1 we guarantee that at least one neighbour has received the original LSU propagating route updates in one direction. We are currently investigating how the ACK_threshold value affects the HSLS's behavior in terms of both overhead and reliability.

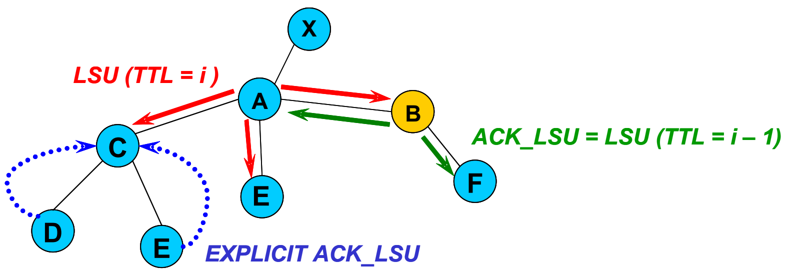


Figure 3.3: ACK_LSU's generation process

3.1.3. HSLS Implementation Architecture

In the following implementation decisions and details of the HSLS software architecture are presented.
Our HSLS module is implemented for the Linux platform due to its open-source nature that allows accessing to OS kernel freely. Furthermore, since the Linux kernel, together with most other parts of the OS, is written in C, we have decided to use the same language; in this way, direct communications such as recovering of network information or frequently interactions with the kernel routing table (e.g. routes' addition and removal) become easier.

HSLS Packets

HSLS utilizes several control packets. It sends routing information over the network using Hello packets during the 1-hop neighbourhood's discovery phase, and LSU packets during the topology dissemination phase. Furthermore, in our HSLS implementation, special packets for the reliability process and packets for the cross-layer interaction must be generated, as well. Our HSLS module uses a unified packet format to flood information in the network. All packets generated by the routing daemon are further encapsulated in UDP datagrams and then sent through the network using UDP connections. As shown in Figure 3.4, the packet is made of a Packet Header, a Message Header and a Message Payload of a variable length. More precisely:
Packet Sequence Number (PSN): (2 byte) it is incremented each time a new HSLS packet is generated and transmitted in the network.
Packet Length (PL): (2 byte) the field stores the total length (in byte) of the packet.
Originator Address (OA): (4 byte) since each node in the network is uniquely identified with an IP address, this field represents the IP address of the node that has generated the packet. This value does not change during the flooding process.
Time to Live (TTL): (1 byte) it contains the maximum number of propagation hops for a packet; each time a node receives a packet it decrements the *TTL* field before broadcasting it to the network; if its value is equal to 0 the forwarding process is stopped.
Packet Type (PT): (1 byte) it indicates which type of packet is encapsulated. Possible values are Hello, LSU, LSU_opt and ACK_expl.
Validity Time (VT): (2 byte) this value indicates how long a node can consider valid the packet information after its reception.
Link Type (LT): (2 byte) it indicates the type of link between the originator node and the advertised neighbours listed after this field. Possible values are symmetric link (SYM) and asymmetric link (ASYM).
Address Size (AS): (2 byte) this field stores the length (in byte) of the list of advertised neighbours that follow a LT field.
Neighbor Address (NA): (4 byte) it represents the main IP address of the advertised neighbor node.
Optional: this field contains information coming from other levels that are not strictly correlated with the routing protocol (i.e. services for middleware).

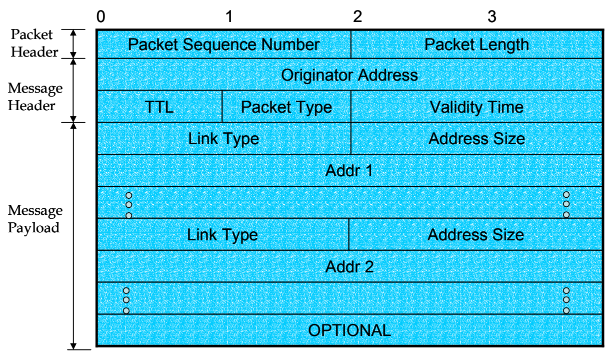


Figure 3.4: Packet format

The Hello packet contains the list of neighbours considering both symmetric and asymmetric links, while only 1-hop neighbours connected through symmetric links are stored in LSU packets. The LSU_opt is an ordinary LSU which also encapsulates in the Optional field extra-data coming from the NeSt. The ACK_exp, used in the reliability process, is made only of Packet and Message Header without any Message Payload.

HSLS Data Structures

The HSLS daemon maintains running state into several information repositories. These data structures are updated during the start-up of the protocol and the processing phase of received control messages; the stored information is used in the generation of messages. Here follows a brief look at the different information repositories used in HSLS.
Interface: this dataset contains internal information of the node (e.g. socket descriptor, name interface, network information).
Topology Table (TT): this repository stores information of all links present in the networks, maintaining also their status (i.e. ASYM/SYM), allowing each node to have knowledge of network topology.
Routing Table (RT): the shortest path and the associated cost to each node of the network are registered here.
In addition, **SentLSUcache (LSC)** and **ReceivedLSUcache (RLC)** are used as repositories of LSU information for a twofold reason: to implement the reliability process (as explained in the previous section), and to detect duplicate packets in order to avoid their processing.
The correct behaviour of the protocol is strictly correlated to these structures; stored information must be always fresh and valid to assure good decisions in routes' calculation and packets' delivery. Hence some data structures' entries have an associated timeout.
More precisely, in TT this value indicates how long the stored information can be considered valid and it is set according to the validity time contained in the packets. The timeout inside the TT is set to the sum of the current time and the validity time, so when the current time is higher than the stored time, the tuple is invalided and its content is not used. In the other two caches this timeout is set to the time window *T* used for the reliability process; after its expiration there will be a retransmission of the same LSU.

HSLS Software Architecture

As shown in Figure 3.5, the software architecture of HSLS protocol is represented by a multi-thread system consisting of several modules running concurrently:
Initialization: it initializes data structures, manages wireless interface and sets socket options.
Packet Management: it generates processes and sends packets in the network.
Garbage Collector: it deletes old entries in the information repositories.
NeSt Communication: this module is used in the cross-layer architecture in order to interface the routing protocol with the NeSt functionalities.

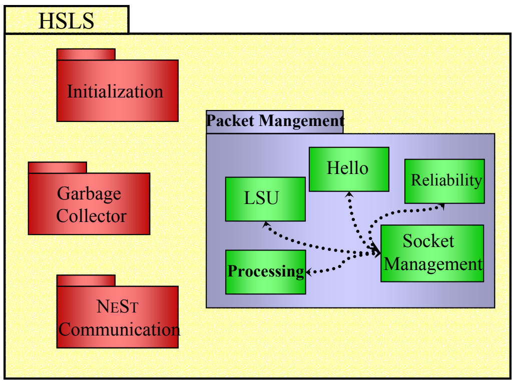


Figure 3.5: Module scheme of the HSLS implementation

Furthermore, the core of HSLS daemon is the Packet Management. It is composed by: modules that generate Hello and LSU packets; a module that checks if some packets must be retransmitted (Reliability); a module which processes packets and updates data structures (Processing) and, last, a module which works directly with UDP socket sending and receiving packets from the network (Socket Management).
In the following a description of modules' interactions is given. Figure 3.6 shows the flows of incoming and outgoing traffic.
Periodically Hello, LSU and Reliability modules generate their own messages according to their procedure and pass them to the Socket Management (event 1). This last module adds Packet Header and/or Optional field with information coming from the NeSt; then the packet is ready to be ejected in the network (2).
The packets' reverse flow is shown in the same picture. All incoming packets are received from the Socket Management (event 3). This module checks if the amount of the received data matches the value of the packet length in the HSLS packet header. If the quantities are equal it passes the packet to the Processing module (4), otherwise it silently discards the packet.

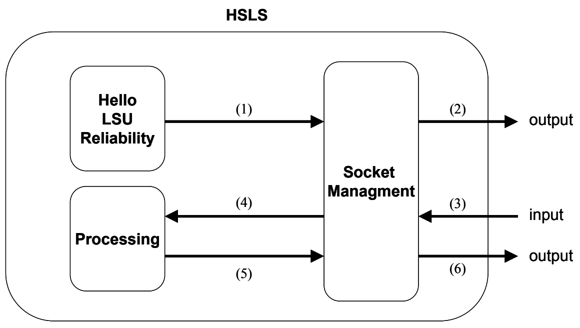


Figure 3.6: Modules interaction

The Processing module first checks (by comparing information stored in the two caches (SLC or RLC)) if the incoming packet is a copy of a previously received one. If it is a duplicate packet, the processing phase ends; otherwise packet's content is used to update the Topology Table. In the latter case, the shortest path tree is computed according to the Dijkstra algorithm and consequently the local Routing Table and the Kernel Routing Table are also updated. If the incoming packet is an LSU in general the Processing module can also i) extract information encapsulated in the Optional field in the case of LSU_opt, and ii) forward the packet according to the forwarding algorithm or create an ACK_expl as previously explained (5). Lastly, Socket Management completes the packet and sends it to the wireless card (6).

As we said, HSLS system runs in threads. This means that there can be simultaneously multiple accesses to the same data structures, for example when a Hello packet is generated by reading information stored in TT, and at the same time an LSU packet is processed causing the update of the same repository. In order to guarantee data integrity, (part of the code of) threads must run in mutual exclusion locking and unlocking shared resources when they are needed; pthread mutex are used with this aim.

3.2. Network support for Services' Location

Service discovery is commonly referred as a middleware service able to lookup characteristics and binding information (like server IP address and port number) of other network services, on behalf of requesting clients.

On fixed networks service discovery relies on multicast messages to find out existing services in the networks. Once the node locates the network entities providing certain services (the servers), they can interact directly with those entities. During service provisioning, servers remain in the same place and can be located through the same route. Thus, in fixed networks or even in mobile networks with limited mobility, the service discovery mechanism can be completely detached from the lower layers. Service discovery scalability in fixed networks is handled using registry servers that maintain service information and are easy discoverable.

The above assumptions clearly do not fit the Ad Hoc paradigm. As mentioned before, one of the main requirements of Ad Hoc networks is *self-organization*. Mobile nodes should be able to self-organize the network, participating to fundamental operations like routing and forwarding. On top of this dynamic infrastructure, nodes may provide application or middleware level services (DHCP servers, DNS servers, NAT, SIP Registrars, 3G Access Point, etc), which turn Ad Hoc networking into a valuable technology for real, general purpose usages. In this scenario, as the networking nodes, as well as other network conditions like nodes density, link status, bandwidth etc., cannot be predicted, also the set of available services will show a high degree of variability. Not only different nodes will support different services every time, but users will be able to shut down their device (and hence the provided services) independently from the rest of the network. Besides, user mobility influences the network topology, determining variable routing conditions, possible (if not frequent) network partitioning, which result in the impossibility of reaching previously available services. In a highly mobile and dynamic scenario such as Ad Hoc networks, service discovery scalability cannot be implemented with common registry servers. Instead, the service discovery would be implemented using a self-collaborative strategy. This strategy requires a different layering structure in order to disseminate service information. Specifically, we propose a cross-layer service discovery mechanism for Ad Hoc networks, which aims at fulfilling the following requirements:

- Decrease the latency associated to service discovery;
- Efficiently update service-provisioning routes;
- Decrease the bandwidth used by service discovery.

3.2.1. Service discovery in Ad Hoc networks

Ad Hoc networks define a new set of requirements that any service discovery mechanism should be compliant with in order to perform an efficient procedure. This includes proper handling of the topology dynamics (i.e. node move and leave/join the network at any time), together with a fair assignment of service discovery tasks to the nodes participating and using the service. This last requirement identifies the *distributed peer-to-peer* programming model, as the one that best suits mobile Ad Hoc environments.

Our proposal adopts a cross-layer mechanism for deploying a network service discovery solution. The approach works toward the stated requirements, and also gives the opportunity for integrations with service discovery solutions present at the application layer. The solution works in conjunction with a link state routing algorithm, proactively spreading around the network short service descriptors, in a reliable and scalable manner. In this way the mechanism promptly reflects situations of newly available services, or services that are no longer reachable. Higher layer modules will access service discovery functionalities through a middleware layer interface, which provides uniform and transparent access to information relative to both local and remote service.

The interaction between the middleware-level service discovery protocol and the link state routing algorithm occurs, in our architecture, through the NeSt (see Section 1.3). Through the NeSt, protocols interact in a loosely coupled manner, following two patterns:
- *Synchronous*, where protocols share internal data.
- *Asynchronous*, where protocols subscribe to and notify events.

Synchronous interactions characterize optimizations on the normal functioning of a protocol. Typically, a protocol P_1 simplifies internal tasks by accessing local information collected and shared by another protocol P_2. Manipulation of shared data happens through the NeSt. Asynchronous interactions characterize the happening of special events to which protocols declared interest. A protocol P_1 subscribes for a specified event *E*, which could be notified either by another protocol P_2 through the NeSt, or by the NeSt itself. The NeSt exports an interface for synchronous and asynchronous interactions, and implements mechanisms to analyze shared data.

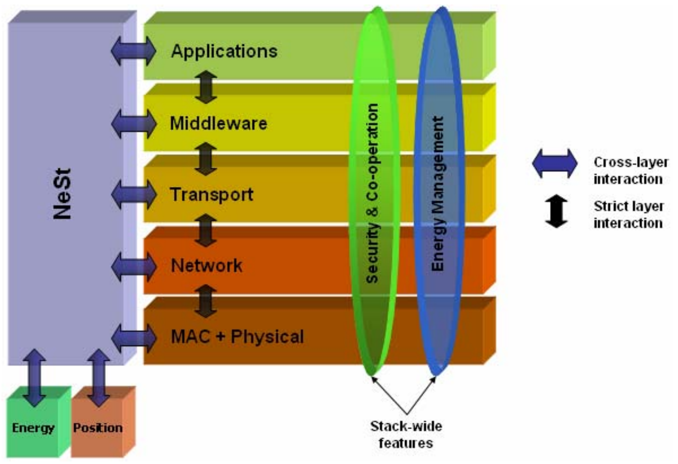


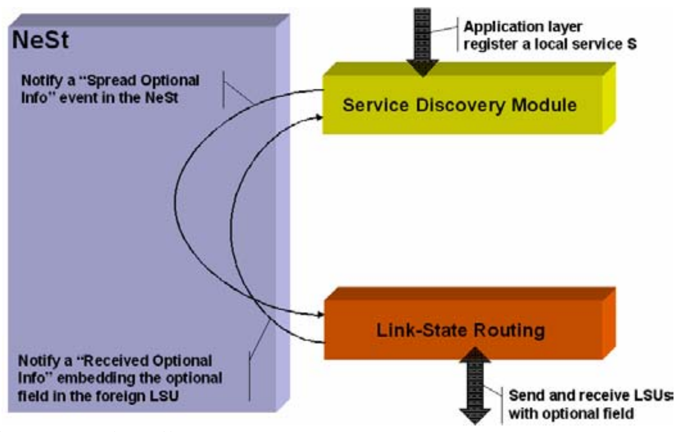
Figure 3.7. Cross-Layer architecture.

3.2.2. Cross-layer service discovery

Our approach to service discovery is based on the concept of cross-layering, where an interaction between protocols at the middleware and routing layers is required for implementing service discovery. In our architecture, a component called Service Discovery Module (SDM) interacts cross-layer with a link-state routing protocol in order to proactively spread data related to locally exported services, inside Link-State Update packets. Additionally, the SDM has the task to collect service related info generated by other nodes and received though incoming updates (i.e., LSU packets). This mechanism eliminates the need for an explicit service discovery protocol, but nevertheless integrates well with existing service discovery mechanisms at the application layer.
In our architecture, the routing protocol plays an important role, as it collects together with information regarding the network topology, the services provided by the network nodes. Sharing this data through the NeSt, simplifies everything that makes use of topology information (e.g. overlay networks, service discovery).
On each Ad Hoc node, the link-state routing protocol continuously updates a local representation of the network topology, as it receives LSUs coming from other nodes. The

fundamental idea is to extend LSU packets, letting them able to carry *optional information* as piggybacked data. A similar approach has been proposed in the literature for extending the *Ad Hoc On-demand Distance Vector* (AODV) routing protocol, to also support the *Service Location Protocol* (SLP). The similarity between the communication patterns used in both AODV and SLP, suggested a merge of the two protocols where *service request* and *reply* messages can be sent in conjunction with *route request* and *reply*.

On each node (see Figure 3.8), a Service Discovery Module (SDM) interacts cross-layer with the link-state routing, notifying its interest in spreading optional information together with subsequent LSU packets. This notification happens through NeSt events. We refer to an event of type "*Spread Optional Info*", as a request to piggyback optional information inside LSU packets, made to the routing protocol by upper layer protocols. Analogously, events of type "*Received Optional Info"* characterize the reception of a link-state update packet enriched with optional information of probable interest for some upper layer entity. In the case of this service discovery architecture, the upper layer entity is the SDM, and the optional data to piggyback and receive are short service descriptors. At bootstrap time, the link-state protocol subscribes for events of type "*Spread Optional Info*", and subsequently receives instances of it as other entities notify the NeSt about occurrences of this event. Finally, the link-state protocol piggybacks the information about the service (contained in the NeSt event instance) inside newly produced LSUs.
In a similar way, the SDM subscribes for events of type "*Received Optional Info*", which the link-state routing notifies when LSUs containing optional information get received from other ad hoc nodes. When such event occurs, the NeSt delivers a copy of the instance, containing the optional content, to all the subscribers. Finally, the SDM checks the optional content, and does the necessary handling in case it contains a short service description.
This mechanism runs distributed across the network without any point of centralization, spreading the workload evenly among the participants. Those nodes running an instance of the SDM generate link-state information enriched with service discovery data, and collect incoming service descriptions, locally building a network service map. The overhead introduced by this cross-layer interaction is well compensated by the elimination of an explicit service discovery protocol. Besides, this distributed mechanism tolerates both node failures and network partitioning, maintaining the scope of the service discovery procedure to the actually available physical network. It is also important to note that this architecture maintains information about available services up-to-date with the routes necessary to reach them. The latency in discovering a particular network service coincides with the propagation of routing information necessary to reach the node offering the service.


NeSt
Notify a "Spread Optional Info" event in the NeSt
Application layer register a local service S
Service Discovery Module
Link-State Routing
Notify a "Received Optional Info" embedding the optional field in the foreign LSU
Send and receive LSUs with optional field
Figure 3.8. Cross-layer service discovery.

The SDM exports an interface to upper layer entities in the protocol stack, allowing for:
- *Registration* of local network services.
- *Lookup* of available services around the network.

The NeSt supports and coordinates the cross-layer interaction, allowing for event subscription, notification and delivery, as described in Section 1.3.

3.2.3. Service access

The proposed cross-layer mechanism allows maintaining the existing application layer service discovery solutions (e.g. UPnP, Jini, SLP, etc). Service information is distributed through the routing layer, collected by the SDM, and can be accessed by upper layer entities through the SDM interface. If an application layer entity looks up the existence of a particular service through the SDM, and receives back an error response (because service data regarding the specified service didn't get spread via the SDM), it may still initiate its own explicit service discovery mechanism (e.g. UPnP, SLP, Jini). For example a UPnP Control Point could initiate its discovery mechanism producing a UPnP M-SEARCH multicast query flooding the network, or instead Control Point send an unicast M-SEARCH if it already knows the IP addresses of existing UPnP devices in the network.

The proposed cross-layer mechanism will be able to handle the mobility of the nodes providing services, performing service discovery and node configuration transparently to upper layers. This guarantees for low latency (minimum delays in discovery services), minimum signaling load, and minimum bandwidth consumption for service discovery transactions.

3.2.4. Service information

In order to not overload link-state information with excessive service data, suitable service descriptors should be defined for piggybacking inside LSUs. These descriptors may consist of integer values that represent unequivocally different services. These descriptors are inserted in the routing messages by events notified by the SDM to the routing module through the NeSt. The SDM receives service related data from the application layer, during a phase of service registration, and periodically formats "*Spread Optional Info*" events containing service information to be spread inside LSU messages. This allows for a combination of the load of doing service discovery with the one of routing, reducing the total overhead accordingly to the size of the ad hoc network.
We define as *Mean Update Time* (MUT), the rate at which the SDM asks the routing for service data spreading, generating a "*Spread Optional Info*" event. Clearly this is a tuning parameter, as lower MUTs generate higher update rates, with an increased traffic of LSUs with service related data. Figure 3.9 shows the extension to routing packets format needed to piggyback service descriptors.

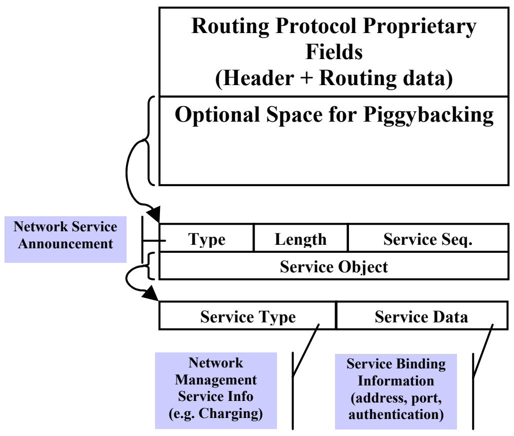

Routing Protocol Proprietary Fields (Header + Routing data)
Optional Space for Piggybacking
Network Service Announcement
Type
Length
Service Seq.
Service Object
Service Type
Service Data
Network Management Service Info (e.g. Charging)
Service Binding Information (address, port, authentication)

Figure 3.9. Data structure extension to enable Network Service Discovery at routing layer.

Service descriptors should implicitly include information about the type of service. For example, when using 32 bits unsigned number for representing services, the 16 first bits could be used for representing the service type (e.g. network management such as DNS, DHCP, NAT, etc or multimedia services such as SIP, UPnP, SLP, etc), while the remaining 16 bits could identify the service itself.

Service descriptors follow a common and well-known packet structure in order to be included as part of the LSU messages (potentially on any link-state or proactive protocol). For example, on a link-sate protocol such as OLSR, they could be included in messages devoted to topology declaration (i.e. TC-messages). Eventually, a list of service descriptors (more than one service on a single node) can be included inside a single OLSR message as shown in Figure 3.10.

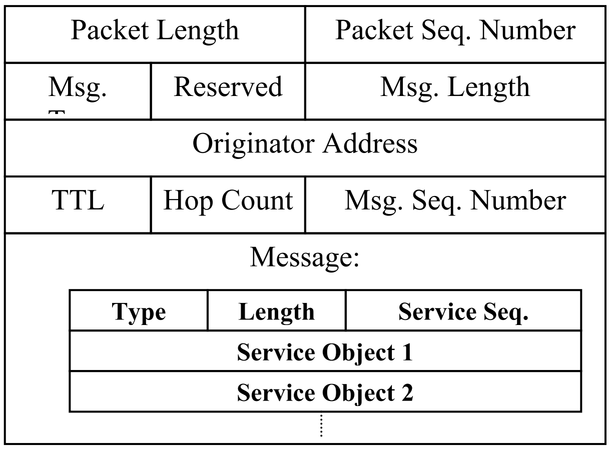


Figure 3.10. OLSR packet with piggybacked service descriptors.

Service descriptors should be also suitable for inclusion in other general-purpose link-state algorithms such as Server Cache Synchronization Protocol (SCSP). SCSP is a generic cache synchronization protocol that can be used for updating link state or service information indistinctly. SCSP provides the flexibility of define different MSU and include service descriptor or link information in a seamless manner. SCSP only considers generic Cache packets (i.e. Cache State Records) that have to be synchronized periodically according to predefined periods. Figure 3.11 shows the type of SCSP messages depending on the information exchanged.

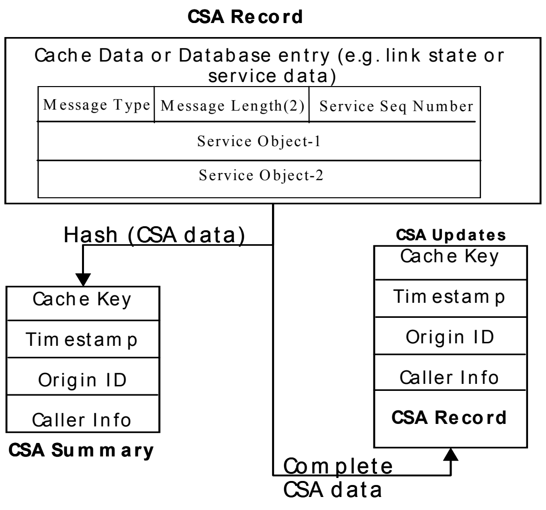
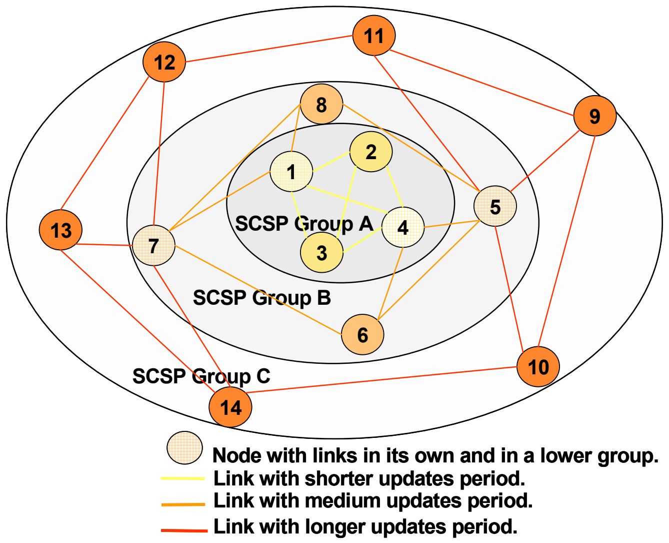


Figure 3.11. SCSP message with inclusion of service descriptors.

When starting SCSP, the synchronization is initiated without exchanging the whole data included in the Cache but instead using a summary (CSA Summary), which uniquely represent the data stored on the Cache Records (CSA Records). Only if the data in the cache has changed then a new packet containing the complete cache data is exchanged (CSA Updates).


SCSP Group A
SCSP Group B
SCSP Group C
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Node with links in its own and in a lower group.
Link with shorter updates period.
Link with medium updates period.
Link with longer updates period.
Figure 3.12. SCSP groups with different updates rate.

The synchronization periods can be modified dynamically and SCSP can consider multiple groups of nodes with different update periods. Therefore, the SD could be included also in the Cache Updates packets used by SCSP without differentiating that it is service or link state information. Figure 3.12 shows an implementation of multiple SCSP groups with different updates rates. In the same network can be nodes participating in multiple SCSP groups and sharing the same link information but with different updates periods. The group in the core of the network have shorter update period in order to maintain a tight neighborhood management. The traffic in this area is higher but localized. Thus, the traffic in the center does not disturb the nodes in the peripheral, which with a lower update rates are still informed about existing services in the center.

3.3. Reliable Forwarding

To achieve well-performing nodes communication for MobileMAN, we designed a mechanism for REliable and Efficient Forwarding (REEF), based on reputation (or reliability), and composed by a trustworthy mechanism for building nodes' reputation, and a set of forwarding policies. The basic idea is to improve performance of the forwarding function, by avoiding unreliable routes, and balancing network utilization at the same time. In our approach, a node is responsible not only for forwarding a packet, but it shall forward it on the route that maximizes its success probability. Goal of REEF is to ensure performance and reliability of data forwarding, in a lightweight manner. The low-power nature of ad hoc devices requires low power consumption: algorithms and mechanisms implementing networking functions should be optimized for lean resource consumption, so as to save capacity for applications, while still providing good communication performance.
The innovative aspect of REEF is that it relies on node's internal knowledge, and does not produce any additional overhead. The reputation building mechanism does not require any neighborhood monitoring or information sharing among nodes. REEF is customized for the MobileMAN *cross layering architecture*. It is positioned at network layer, and exploits routing information, as well as transport layer packet acknowledgements. Specifically, REEF assumes that i) TCP acknowledgments are available also to the other layers in the protocols' stack; ii) a pro-active link-state routing protocol (with partial information dissemination) is

used. Such information enables the application of forwarding policies that aim at maximizing route's success and at balancing network traffic.

As already presented in [9], every node has a dynamically updated performability table containing a value for every outgoing link to a neighbor. Such a value represents a performability index for paths rooted at that neighbor. Every time the node sends a packet on a path, it updates the performability value associated to the neighbor through whom the packet has passed: the updating is positive whenever source node receives an acknowledgement from destination, negative otherwise. In particular, if the node *s* sends a packet to the node *d* through the neighbor *j*, and M is the result of the packet delivery, then the reliability index of the node *j* is updated in the following way:

$$R_i \leftarrow \alpha \cdot R_i + (1-\alpha)M$$

where α, 0≤ α ≤ 1, is a smoothing factor and represents the percentage of the previous estimate considered on each new estimate. If α =0.9, then ninety percent of the new estimate is from the previous estimate, and 10% is from the new measurement. M is the result of a packet delivery process from s to d, and it can assume the following values:

$$M = \begin{cases} 0 & \text{if } s \text{ does not receive ack from } d \\ 1 & \text{if } s \quad \text{receives ack from } d \end{cases}$$

Performability indices are then used to control traffic forwarding. In case of multiple routes available for packet forwarding to a destination node, the source node can choose one of them according to a certain principle. Typical ad hoc policies consider shorter or fresher paths more desirable than others. The drawback of such criteria is twofold: i) some area of the network are more prone to high traffic load (network hot-spots), ii) route reliability is not considered. Hereafter, we show that policies taking into account reliability improve network performance. Assuming the existence of multiple paths to a destination we investigate the effectiveness of the following policies:[13]

Best-Route Source node takes always the most reliable route. In such a case, source node compares performability values for available routes and forwards packets on the link with the greatest value. This policy assures source node of taking always the most reliable route. The main drawback of such a choice is the deviation of all traffic on most reliable links which, in case of high traffic load, can quickly get congested.

Probabilistic This policy relates performability values of available routes to build a probabilistic scheme. Let us suppose we have several possible routes to a destination through different source's neighbor nodes, $i_1, i_2, \ldots, i_n$. Each neighbor has its respective performability value $R_{i_1}, R_{i_2}, \ldots, R_{i_n}$. We associate a probabilistic value to each of such neighbors, $p_{i_j}, 1 \le j \le n$, defined in the following way:

$$p_{i_j} = \frac{R_{i_j}}{\sum_{K=1}^{n} R_{i_k}}$$

The last policy relates *performability* values so that the resulting probabilistic value reflects the link performability level. Routes are chosen according to the probabilistic value associated to the first node on the path: the greater the probability, the higher the route selection frequency. This probabilistic policy allows nodes to take even less reliable routes: traffic forwarding function is better distributed on all available routes and links congestion becomes rarer.

13 The selected policy are quite simple, the reliable forwarding mechanism can be applied also to other, more complex, policies.

3.3.1. Evaluation

The objective of this evaluation is to test the effectiveness of the policies for route selection. Specifically, we want to investigate the throughput achieved by applying different forwarding policies. As shown in `Figure 3.13`, the simulated network is composed by a source node S, a destination node D, and three different routes connecting them. On the first and third route, all nodes are cooperative, while on the second one, there is a selfish node discarding packets according to a selfishness model (which will be defined later). Every time the node S sends a packet on one of the three paths, it updates the reliability value associated to the neighbor through which the packet was forwarded[14]. To evaluate network performance for different route-selection policies, we defined four scenarios and conducted experiments applying different policies to such scenarios.

- **Scenario 1** - All links have the same transmission speed (2 Mbps). Nodes on *route 1* and *3* are all cooperative, so that there is no packets dropping, while there is a selfish node in *route 2* (M), that provokes packets' losses with probability *p=0.7*.
- **Scenario 2** - The only difference with the previous scenario is the selfishness model for node M: it cooperates for 500 packets, and then goes in a stand-by state, discarding the next 500 packets.
- **Scenario 3** - We varied transmission speed of links on *route 1* and *3* so as to cause possible congestion. Specifically, on nodes Y and J transmission speed passes from 2 Mbps (incoming link) to 1 Mbps (outgoing link); both nodes have buffer capacity of 10 packets. Consequently, both nodes Y and J can get congested and drop arriving packets while the buffer is full. Node M is selfish, as described in scenario 2.
- **Scenario 4** - We varied links transmission speed increasing the possibility for a bottleneck on *route 1*. Specifically, on node Y (*route 1*) transmission speed passes from 2 Mbps (incoming link) to 0.5 Mbps (outgoing link), while on node J (*route 3*) transmission speed passes from 2 Mbps to 1.5 Mbps. Buffer capacity on both nodes is 10 packets. Node M is selfish, as described in scenario 2.

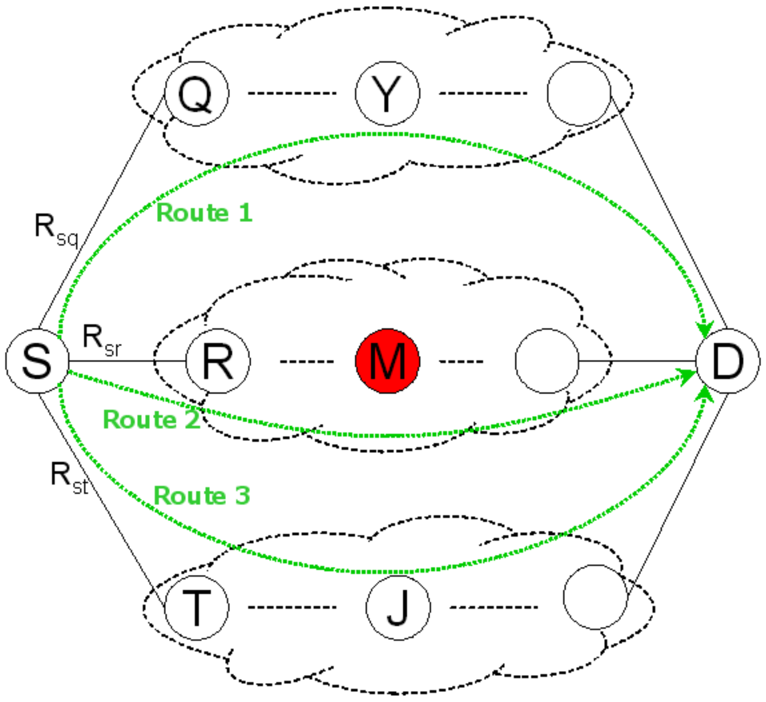


Figure 3.13. In the simulated network, node S has three different routes to reach node

14 For the sake of simplicity, simulations implement immediate acknowledgments of delivered packets and no loss of ACKs. This choice does not affect the meaning of obtained results, and effects of ACK delay on reliability estimates is currently under study.

D. Node M on route 2 is selfish.

Experiments have been performed in each scenario by applying the different route selection policies. We simulated also the conventional case in which the forwarding node does not consider reliability indexes, and distributes packets equally on the three routes. Choosing each route with the same probability represents the best compromise when no selection criterion[15] is used, because it minimizes congestion events. In the following, we call such a criterion of traffic distribution *load-balancing*, to indicate that reliability values are not considered, but traffic is equally spread on each route.

We evaluated the model with the same traffic load, and observed network throughput. Packets inter-arrival times are exponentially distributed (i.e. generated traffic follows a Poisson's model). Simulations have been performed with independent replications technique (number of replication set to 5). In particular, we measured offered load (OL) and throughput (γ), indicating also the confidence interval.
The confidence level is 95%, and the α parameter to update reliability is set to 0.9. Results are shown in `Figure 3.14` and `Figure 3.15`.

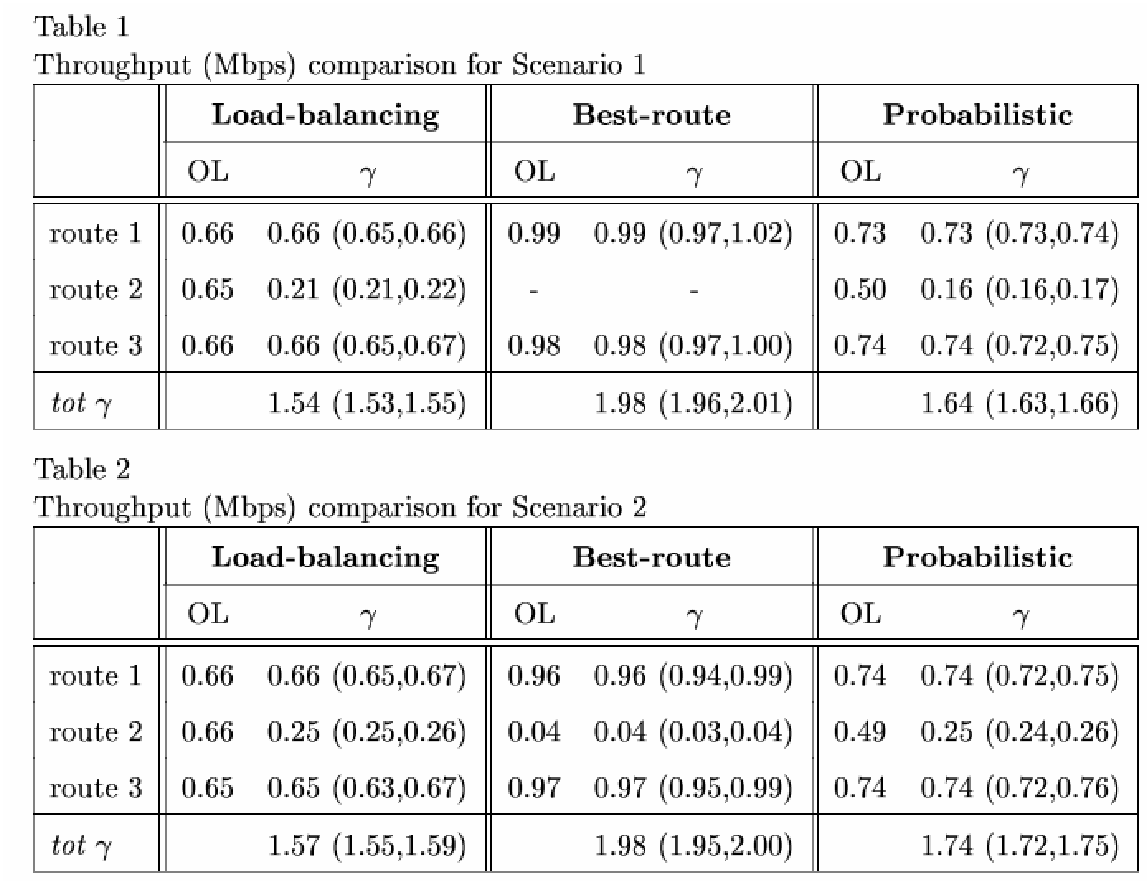


Table 1
Throughput (Mbps) comparison for Scenario 1

	Load-balancing		Best-route		Probabilistic	
	OL	γ	OL	γ	OL	γ
route 1	0.66	0.66 (0.65,0.66)	0.99	0.99 (0.97,1.02)	0.73	0.73 (0.73,0.74)
route 2	0.65	0.21 (0.21,0.22)	-	-	0.50	0.16 (0.16,0.17)
route 3	0.66	0.66 (0.65,0.67)	0.98	0.98 (0.97,1.00)	0.74	0.74 (0.72,0.75)
tot γ		1.54 (1.53,1.55)		1.98 (1.96,2.01)		1.64 (1.63,1.66)

Table 2
Throughput (Mbps) comparison for Scenario 2

	Load-balancing		Best-route		Probabilistic	
	OL	γ	OL	γ	OL	γ
route 1	0.66	0.66 (0.65,0.67)	0.96	0.96 (0.94,0.99)	0.74	0.74 (0.72,0.75)
route 2	0.66	0.25 (0.25,0.26)	0.04	0.04 (0.03,0.04)	0.49	0.25 (0.24,0.26)
route 3	0.65	0.65 (0.63,0.67)	0.97	0.97 (0.95,0.99)	0.74	0.74 (0.72,0.76)
tot γ		1.57 (1.55,1.59)		1.98 (1.95,2.00)		1.74 (1.72,1.75)

Figure 3.14. Throughput comparison for Scenario 1 and 2.

In the first and second scenario (see Tables in `Figure 3.14`), the best-route policy outperforms all the others. Achieved throughput is close to the maximum value (2 Mbps) and equally distributed among the first and the third route. As soon as the node M starts discarding packets, the reliability value in that direction decreases, and the source node does not consider anymore that path because it has lower reliability than the others. However, the probabilistic policy achieves good throughput as well, and performs better than load-

[15] Shorter routes are generally preferred.

balancing. While with the latter the OL is equally distributed among available routes, the probabilistic policy sends more traffic on the first and the third route (about 0.74) than on the second one (about 0.50). In comparison with best-route, the probabilistic policy has the advantage of not totally excluding the second path (which contains the selfish node), but chooses it less frequently than others, distributing traffic more fairly, at the expense of throughput. It is worth noticing that the node on the second route could be congested instead of selfish. In this case, the probabilistic policy would frequently test the reliability on the second route, and hence would start increasing its usage as congestion fades away.

In the third scenario we investigated effects of congestion, in addition to selfishness. In experiments concerning this scenario, both routes 1 and 3 contain nodes that may get congested in case of high traffic load, as they have a limited buffer capacity, and link speed passes from 2 Mbps (incoming link) to 1 Mbps (outgoing link). In this case, the gap between the best-route and the probabilistic policy is small and both outperform the load-balancing, which does not cause congestion but offer too much load to route 2 (OL=0.66 while γ=0.25). The best-route policy has a total throughput of 1.76 but causes congestion, with measured loss probability of about 0.1, on both route 1 and 3 (γ is less than OL). On the other side, the probabilistic policy achieves an analogous result (total γ=1.73) without causing congestion, because it exploits also route 2 (where OL=0.5 and γ=0.25).

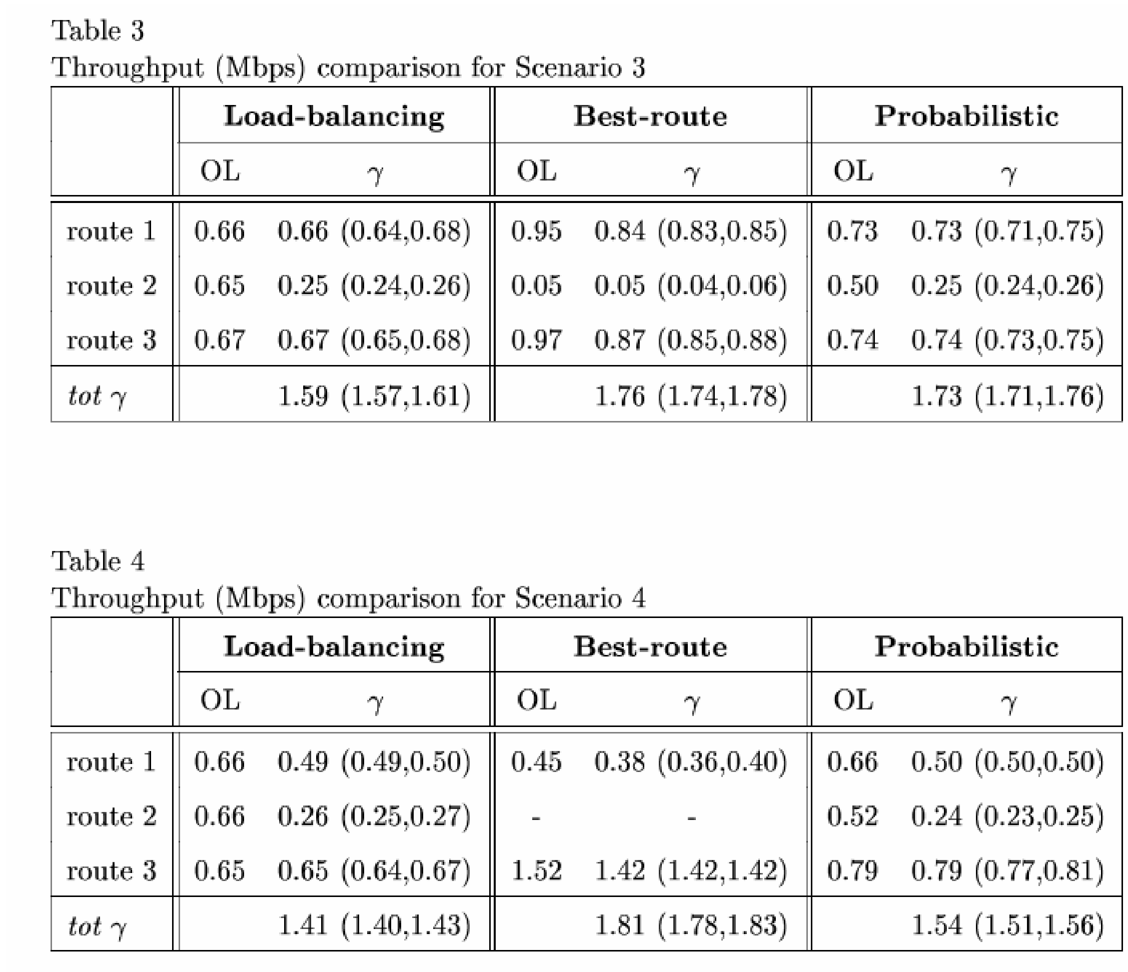


Table 3
Throughput (Mbps) comparison for Scenario 3

	Load-balancing		Best-route		Probabilistic	
	OL	γ	OL	γ	OL	γ
route 1	0.66	0.66 (0.64,0.68)	0.95	0.84 (0.83,0.85)	0.73	0.73 (0.71,0.75)
route 2	0.65	0.25 (0.24,0.26)	0.05	0.05 (0.04,0.06)	0.50	0.25 (0.24,0.26)
route 3	0.67	0.67 (0.65,0.68)	0.97	0.87 (0.85,0.88)	0.74	0.74 (0.73,0.75)
tot γ		1.59 (1.57,1.61)		1.76 (1.74,1.78)		1.73 (1.71,1.76)

Table 4
Throughput (Mbps) comparison for Scenario 4

	Load-balancing		Best-route		Probabilistic	
	OL	γ	OL	γ	OL	γ
route 1	0.66	0.49 (0.49,0.50)	0.45	0.38 (0.36,0.40)	0.66	0.50 (0.50,0.50)
route 2	0.66	0.26 (0.25,0.27)	-	-	0.52	0.24 (0.23,0.25)
route 3	0.65	0.65 (0.64,0.67)	1.52	1.42 (1.42,1.42)	0.79	0.79 (0.77,0.81)
tot γ		1.41 (1.40,1.43)		1.81 (1.78,1.83)		1.54 (1.51,1.56)

Figure 3.15. Throughput comparison for Scenario 3 and 4.

In scenario 4, we varied links transmission speed, so as to cause different cases of congestion. In this scenario, the best-route policy performs better than others, as the offered load on both routes 1 and 3 is very close to the respective link capacity: on route 1 OL is 0.45 with γ=0.38

(loss probability=0.15), while on route 3 OL is 1.52 with γ=1.42 (loss probability=0.06). The probabilistic policy improves load-balancing by increasing the OL on route 3 compared with route 2, but does not avoid congestion on route 1.

To summarize, both reliability-aware forwarding policies perform always better than load-balancing. In some cases, the best route policy outperforms the probabilistic one. This happens especially in scenario 4 that is characterized by selfishness as well as congestion events (scenarios 1 and 2 are less realistic because do not experience congestion events). On the other side, in scenario 3 the probabilistic and best-route policies achieve similar results. This is because the probabilistic choice does not cause congestion, and there is balance between what it achieves on route 2 and what the best-route policy gains on route 1 and 3. Future work will investigate the possibility of combining the advantages of these two policies in order to further improve throughput and reduce congestion.

3.4. TPA: A Transport Protocol for Ad hoc Networks

Several papers have pointed out that the TCP behavior in a multi-hop ad hoc network is far from ideal, see [28]. Many aspects contribute to this non-ideal behavior. They are discussed in detail in Section 3.4.1. To improve the performance of the TCP protocol in multi-hop ad hoc networks several proposals have been presented ([12], [13], [14], [15], [16], [17], [26]). Almost all these proposals are modified versions of the legacy TCP protocol. However, as explained in Section 3.4.1, MANET behaves in a completely different way from wired networks (e.g., Internet) for which the TCP protocol was originally conceived. Therefore, we think that it is more fruitful to think in terms of a new transport protocol optimized for MANETs rather than adapting the TCP protocol to the ad hoc environment. In this framework, interoperability with fixed-Internet computers may be achieved by exploiting the Indirect-TCP model [23]. The authors in [26] take a similar approach.
Hereafter we present a novel transport protocol, named TPA (Transport Protocol for Ad hoc networks) specifically tailored to the characteristics of the MANET environment. It provides a reliable, connection-oriented type of service and includes several innovations with respect to the legacy TCP protocol. Specifically, the TPA protocol is able to manage route failures and route changes that may arise due to nodes' mobility. In addition, the congestion control mechanism is completely re-designed with respect to the legacy TCP. Finally, TPA implements a novel retransmission policy to reduce the number of useless retransmissions and, hence, the power consumption.
We evaluated the TPA protocol by simulation and compared it with the TCP protocol. To decouple the effects on TCP of congestion and nodes’ mobility, we only considered static nodes. The results obtained show that, even in a static scenario, TPA outperforms TCP in all operating conditions. In every experiment that we run, the throughput achieved by the TPA is never less than the throughput achieved by the TCP. Interestingly, the TPA always use a lower number of retransmissions, and this results in power saving. For example, in the case of high congestion, the number of retransmissions required by the TCP is between twice and sixteen times the number of retransmissions required by the TPA. Furthermore, in the case of high congestion, the throughput achieved by TPA is around twice the throughput achieved by TCP.
The difference in performance is expected to be even greater in a mobile scenario because, unlike TCP, TPA includes mechanisms to manage efficiently route failures and route changes caused by nodes’ movements.

In the following we first highlight the different behavior of MANETs with respects to traditional wired networks for which the TCP protocol was designed, and discusses motivations for designing a novel transport protocol (see Section 3.4.1). In Section 3.4.2 we describe the TPA protocol, while in Section 3.4.3 we present the TPA performance evaluation.

3.4.1. Motivations for a Novel Transport Protocol

TCP was originally conceived for wired networks, like the Internet, where nodes do not change their position over time. Furthermore, it assumes that packet losses are almost always due to congestion phenomena causing buffer overflows at intermediate routers. The above assumptions do not hold in MANETs. The MANET topology may change very frequently due to nodes' mobility. In addition, nodes may fail due to battery exhaustion. Finally, congestion phenomena as intended in the Internet are rare events in MANETs, since packet losses due to link-layer contentions are largely predominant [18].

Nodes' movements and failures cause phenomena like *link failures, route failures, route changes* and sometimes *disconnections*. In particular, a link failure occurs every time a node A fails or changes its position so that a neighboring node B is no longer able to communicate with it. As a side effect, the link between A and B is broken and all connections including this link experience a link failure. From the point of view of a connection, a link failure may lead to a route change (if node B can found an alternative route towards the final destination) or a route failure followed by a route change (if node B cannot find an alternate route). Route failures and route changes usually produce packet losses that manifest themselves as duplicated ACKs or timeout expirations at the sender. Even when a route change does not produce packet losses (e.g., the alternative route is immediately available) it usually results in a different Round Trip Time (RTT) experienced by packets. This may lead to timeout expirations at the sender. The TCP is not able to manage route failures and route changes efficiently. The sender TCP misinterprets duplicated ACKs and timeouts due to route failures or route changes as congestion and activates the congestion control mechanism. This leads to both unnecessary retransmissions (in the case of a route failure, the source TCP retransmits lost segments even if no route to the destination is available), and throughput degradation (when a new route is found, the TCP typically starts from the slow-start phase) ([12], [13], [19]). Of course, unnecessary packets retransmissions consume energy both at the sending and intermediate nodes.

Even assuming that nodes in the MANET are static, the MANET behavior is significantly different from that of a traditional wired network. Specifically, packet losses are originated by different phenomena. In traditional wired networks, like the Internet, packet losses are almost always due to congestion phenomena causing buffer overflows at intermediate routers. That is not true in MANETs where buffer overflows at intermediate nodes are rare events, while packet losses due to *link-layer* contentions predominate (packet losses due to transmission errors are recovered through link-layer retransmissions). In other words, congestions manifest themselves as link-layer contentions rather than as buffer overflows. For instance, in IEEE 802.11-based ad hoc networks, packets are dropped primarily because of hidden and exposed node problems causing a node to back off for seven consecutive times and discarding the packet. On the other hand, buffer overflows may never occur unless node buffers are extremely small [18].

The TCP protocol reacts to packet losses originated by link-layer contentions by activating the legacy congestion-control mechanism. A direct consequence is that the TCP window size is typically far from the optimal value, i.e., the value for which the connection throughput is maximized ([19], [20]). The optimal window size depends on the number of hops. In ([18], [19]) it is shown that in few-hop connections TCP achieves the best performance when using a small window size (i.e., 2-4 packets). On the other side, several studies have shown that, in practice, the number of hops in a MANET connection is expected to be typically small. In [21] it is shown that the throughput experienced by a connection on a MANET decays as $1/\sqrt{n}$, where n is the number of hops. A similar result has been obtained through experimental measurements performed on IEEE 802.11 ad hoc networks in [22], where it is shown that the connection throughput decays as $1/n^{1.68}$. From these results it follows that in real MANETs it is reasonable to expect connections spanning only few hops. Therefore the TCP optimal window size would be around 2-4 packets. However, the TCP congestion control mechanism doesn't stabilize the window size at its optimal value and allows the

window size to grow beyond this optimal value. This behavior produces link-layer congestions at intermediate nodes and, hence, throughput degradations.
From the above remarks it clearly emerges that the TCP protocol is not suitable for MANETs. It also emerges that MANETs behave in a completely different way from traditional wired networks for which the TCP was conceived. Therefore, we think that it is better to design a new transport protocol, specifically tailored to the MANET characteristics, rather adapting the TCP protocol to the ad hoc environment. The new protocol should be able to detect and manage efficiently route failures and route changes caused by nodes' movements. Specifically, it should avoid unnecessary retransmission (e.g., after a route failure) to save energy. Finally, the congestion control mechanisms should be designed by taking into account the real nature of congestion phenomena in MANETs. Specifically, the maximum window size should be small.

One may argue that the TCP is currently used in millions of nodes connected to the Internet. However, using a single TCP connection on a hybrid internet including a MANET and a wired network would experience low performance [20]. The two networks have contrasting requirements in terms of window size (in wired networks the optimal window size is usually larger than few packets), and choosing an intermediate value may not be a good solution [20]. To overcome this drawback, the Indirect-TCP model [23] could be used.

3.4.2. TPA Protocol Description

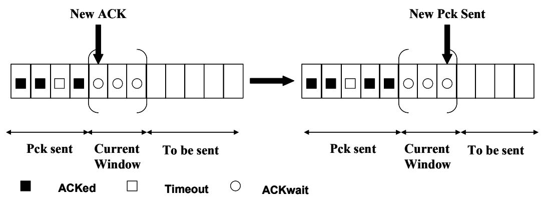
The TPA protocol provides a reliable, connection-oriented type of service. The set up and tear down phases are similar to the corresponding phases in the TCP protocol, and are thus omitted for the sake of space. In the following we focus on the data transfer phase.

3.4.2.1. Data Transfer

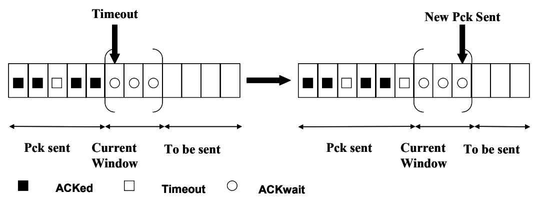
The TPA protocol is based on a sliding-window scheme where the window size varies dynamically according to the flow control and the congestion control algorithms. The congestion control mechanism is described hereafter, while the flow control mechanism is similar to the corresponding TCP mechanism [24] and is thus omitted. The TPA tries to minimize the number of (re)transmissions in order to save energy. To this end, packets to be transmitted are managed in blocks, with a block consisting of *K* packets[16]. The source TPA grabs a number of bytes - corresponding to *K* TPA packets - from the transmit buffer[17], encapsulates these bytes into TPA packets, and tries to transmit them reliably to the destination. Only when all packets belonging to a block have been acknowledged the TPA takes care to manage the next block. Each packet header includes a *sequence number* field that identifies the block to which the packet belongs, and a *data_bitmap* field consisting of *K* bits to identify the position of the packet within the block. The TPA header also includes two fields for piggybacking ACKs into data packets: *acknowledgement number* and *ack_bitmap*. The *acknowledgement number* identifies the block containing the packet(s) to be acknowledged, while a bit set in the *ack_bitmap* indicates that the corresponding packet within the block has been received correctly by the destination. Of course, it is possible to acknowledge more than one packet by setting the corresponding bits in the bitmap (a single ACK contains information for all the packets within the block).
Packet transmissions are handled as follows. Whenever sending a packet, the source TPA sets a timer and waits for the related ACK from the destination. Upon receiving an ACK for an outstanding packet the source TPA performs the following steps: i) derives the new window size according to the congestion and flow control algorithms; ii) computes how many packets can be sent according to the new window size; and iii) sends next packets in the block (see `Figure 3.16a`). On the other hand, whenever a timeout related to a packet in the current window expires, the source TPA marks the packet as "timed out" and executes steps i)-iii) as

[16] TPA packets have the same size of TCP segments.
[17] A block may include less than *K* packets if the buffer does contain a sufficient number of bytes.

above, *just as in the case the packet was acknowledged*[18] (see `Figure 3.16b`). In other words, the TPA performs a transmission round during which it sends all packets within the block, without retransmitting timed-out packets. Then, the sender performs a second round for retransmitting timed-out packets, which are said to form a "retransmission stream" (see `Figure 3.17`). In the second round the sender performs steps i)-iii) described above with reference to the retransmission stream instead of the original block. This procedure is repeated until all packets within the original block have been acknowledged by the destination. If an ACK is received for a packet belonging to the retransmission stream, that packet is immediately dropped from the stream.
The proposed scheme has several advantages with respect to the retransmission scheme used in the TCP. First, the probability of useless retransmissions is reduced since packets for which the ACK is not received before the timeout expiration are not retransmitted immediately (as in the TCP protocol) but in the next transmission round. This is particularly important in MANETs where nodes are highly mobile and, thus, the timeout value might not reflect the current RTT of the connection (see also Section 3.4.2.2). It should also be observed that the longer waiting time in the TPA protocol does not result in throughput degradation, since during this time interval the sender transmits other packets. Second, the TPA is resilient against ACK losses because a single ACK is sufficient to notify the sender about all missed packets in the current block. Third, the sender does not suffer from the out-of-order arrivals of packets. This implies that the TPA can operate efficiently also in MANETs using multi-path forwarding [25]. On the other hand, using the TCP in such networks might result in several duplicated ACKs at the sender side, thus activating the congestion control mechanism with the consequence of throughput degradation.



(a)



(b)

Figure 3.16. Actions performed by the TPA sender upon receiving an ACK (a), or when a timeout occurs (b).

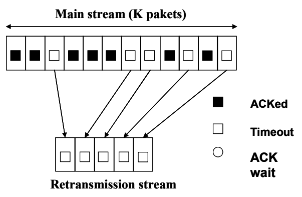


Figure 3.17. Retransmission Stream.

TPA also includes mechanisms to adapt dynamically to varying network conditions. Specifically, it is able to detect and manage three kinds of events: *route failures*, *route changes* and *congestions*.

3.4.2.2. Route Failure Management

Like many other solutions ([12], [13], [14], [15]), the TPA protocol can exploit the network-layer support, if available, for detecting route failures ([10], [11]). Whenever an intermediate node realizes that a packet cannot be forwarded to the next node because of a link failure, and no alternative route to the destination is available, it may send an *Explicit Link Failure*

[18] Due to the congestion control algorithm, in this case the window size might be shrunk after executing point i).

Notification (ELFN) back to the sender node. Upon receiving an ELFN, the source TPA enters a *freeze* state where the transmission window size is limited to one packet. To limit the number of packets sent when there is no available route, while in the freeze state TPA sends new packets with a retransmission timer that is doubled after each expiration (i.e., the same algorithm is used in the TCP protocol [24]).
However, even if the underlying layer does not provide the ELFN service, the sender TPA is still able to detect route failures as it experiences a number of consecutive timeout. Specifically, the sender TPA assumes that a route failure has occurred whenever it detects th_{ROUTE} consecutive timeouts. In this case it enters the freeze state. In the freeze state, the TPA sender behaves as described above. Obviously, th_{ROUTE} is a protocol parameter that needs to be set appropriately.

We assume that the network layer does not provide route re-establishment notifications. Therefore, TPA realizes that the route has been re-established as soon as it receives an ACK for the latest packet sent. Upon reception of such an ACK, the TPA i) leaves the freeze state; ii) sets the congestion window to the maximum value *CWNDmax*; and iii) starts sending new packets (the *rtr* parameter is set to 0). On the other hand, if route re-establishment messages are available, the TPA behavior is further optimized. Specifically, in the freeze state the TPA refrains from transmitting any packet, waiting for a route re-establishment message.

3.4.2.3. Route Change Management

Similarly to TCP, TPA estimates the RTT of the connection, and then, uses this estimate to set the retransmission timeout RTO. Both parameters are derived in the same way as in the TCP protocol, as follows:

$$\begin{aligned} &ERTT_{rtt}(n) = g \times RTT(n) + (1-g) \times ERTT_{rtt}(n-1) \\ &DEV_{rtt}(n) = h \times \left| RTT(n) - ERTT_{rtt}(n) \right| \times (1-h) \times DEV_{rtt}(n-1) \\ &RTO(n) = ERTT_{rtt}(n) + 4 \times DEV_{rtt}(n) \end{aligned},$$

where: i) $ERTT_{rtt}(n)$ and $DEV_{rtt}(n)$ are the average value and standard deviation of the RTT estimated at the n^{th} step, respectively; ii) *RTT(n)* denotes the n^{th} RTT sample; iii) *RTO(n)* is the retransmission timeout computed at the n^{th} step; and iv) g and h ($0 < g, h < 1$) are real parameters (see [24] for details).
Whenever a route change occurs, the new path may differ from the previous one in terms of number of hops. This means that, after a route change, packets may experience a variation in the RTT and the re-transmission timeout might be no longer appropriate for the new path. To avoid possible re-transmissions, the TPA protocol must detect route changes as soon as they occur, and modify the RTT estimation method in such a way to achieve quickly a reliable estimate for the new RTT. In practice, TPA detects that a route change has occurred either i) when a new route becomes available after a route failure; or ii) when th_{RC} consecutive samples of the RTT are found to be external to the interval $[ERTT_{rtt} - DEV_{rtt}, ERTT_{rtt} + DEV_{rtt}]$.
Upon detecting a route change, the TPA replaces the g and h values in the *ERTT* and *DEV* estimators to greater values (*g1* and *h1*) so that the new RTT estimates is heavily influenced by the new RTT sample. This allows achieving a reliable estimate of the new RTT immediately after the route change has been detected. Finally, after n_{RC} updates of the estimated RTT, the parameter values are restored to the normal g and h values.

3.4.2.4. Congestion Control Mechanism

Congestions due to link-layer contentions manifest themselves at the transport layer in two different ways. An intermediate node may fail in relaying *data* packets to its neighboring nodes and, thus, it sends an ELFN back to the sender node (provided that this service is supported by the network layer). This case, throughout referred to as *data inhibition*, cannot be distinguished by the sender TPA from a real route failure. On the other hand, an intermediate node may fail in relaying *ACK* packets. In this case, throughout referred to as

ACK inhibition, the ELFN (if available) is received by the destination node (i.e., the node that sent the ACK), while the source node (i.e., the node sending data packets) only experiences consecutive timeouts. Whenever the sender TPA detects th_{CONG} consecutive timeout expirations it assumes that an *ACK inhibition* has occurred, and enters the *congested* state. The source TPA leaves the congested state as soon as it receives th_{ACK} consecutive ACKs from the destination.
If the network layer does not support the ELFN service, the only way to detect both data and ACK inhibitions is by detecting consecutive timeouts at the sender. Congestions and route failures are no longer distinguishable. Hence, th_{CONG} and th_{ROUTE} collapse in the same parameter, and the freeze and the congested state collapse in the same state.

The TPA congestion control mechanism is window-based as in the TCP protocol. However, as anticipated, in the TPA the maximum congestion window size (*CWNDmax*) is very small (in the order of 2-3 TPA packets) and, hence, the maximum and minimum values are very close. Therefore, the TPA congestion control algorithm is very simple. In normal operating conditions, i.e., when the TPA is not in the congested state, the congestion window is set to the maximum value, *CWNDmax*. When the TPA enters the congested state, the congestion window is reduced to 1 to allow congestion to disappear.

3.4.3. Simulation Analysis

In this section we compare, by simulation, the performance of TCP and TPA. To this end, we developed a custom simulation model that extends the model used in [27]. Our simulator includes the IEEE 802.11 MAC protocol and physical channel model, the DSR routing protocol, and the transport protocol (TCP or TPA). To be aligned with existing literature, we assumed the physical channel model characterized by parameter values defined in TABLE I and TABLE II.

TABLE I. Physical channel model.

Parameter	**Value**
Bit rate	2 Mbps
Transmission range	376 m
Interference range	676 m
Carrier sensing range	676 m

TABLE II. Operational Parameters.

Parameter	**Value**
Distance between node	300 m
Packet Size (TCP/TPA)	512 Bytes
CBR Packet Size (UDP)	512 Bytes
th_{ROUTE} (TPA)	3
th_{ACK} (TPA)	1
Block Size (TPA)	12

To exercise the TCP protocol in a favourable environment we considered a static scenario (i.e., absence of node mobility). Furthermore, we assumed that the network layer provides neither the ELFN nor the route re-establishment services.
In our experiments we considered the string topology depicted in Figure 3.18, where the distance between consecutive nodes is 300 m. Therefore, each node is within the transmission range of only adjacent nodes, and the carrier sensing and interference ranges span two hops (for instance, when node 4 in Figure 3.18. is transmitting, nodes 2 and 6 can hear its transmission). According to the remarks in previous sections, in our experiments we considered a TCP/TPA connection that spans a limited number of hops. We assumed that node 1 is sending ftp-like traffic to node 6. In addition, to investigate the effects of background traffic on the performance of the TCP/TPA connection, we also considered a CBR (Continuous Bit Rate) session where node 2 sends periodically (UDP) packets to node 5. We compared the performance of TCP and TPA protocols both in terms of *throughput* achieved by the destination node at the application layer, and *percentage of retransmission*, i.e., the percentage of packets retransmitted by the TCP/TPA sender. Since (re-)transmissions

consume energy (both at the sender and intermediate nodes) the percentage of retransmissions can be regarded as an energy-efficiency index.

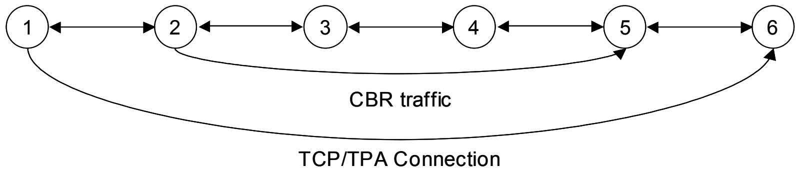
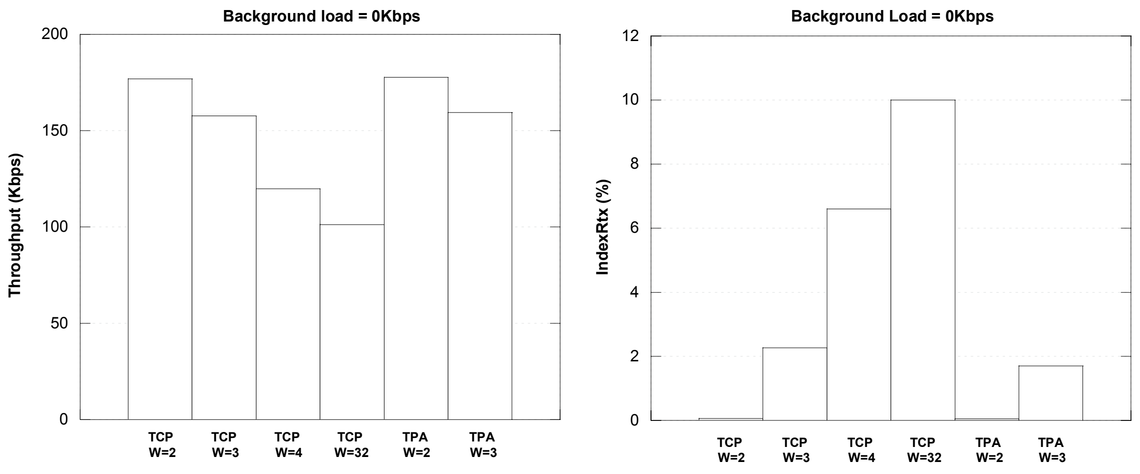


Figure 3.18. Network Configuration.

3.4.3.1. Simulation Results

As a preliminary step in our analysis, we exercised TCP and TPA in the network scenario described above to determine the optimal maximum window size. In these experiments we did not consider any background traffic. The results obtained are summarized in Figure 3.19. It clearly appears that the optimal window size for both protocols is 2. For the TCP protocol this result is aligned with the analysis in ([18], [19]). In this scenario both protocols exhibit similar performance both in terms of throughput and retransmissions. The reason is that in a static MANET without any interfering traffic the TCP with limited maximum window size exhibits good performance [19].



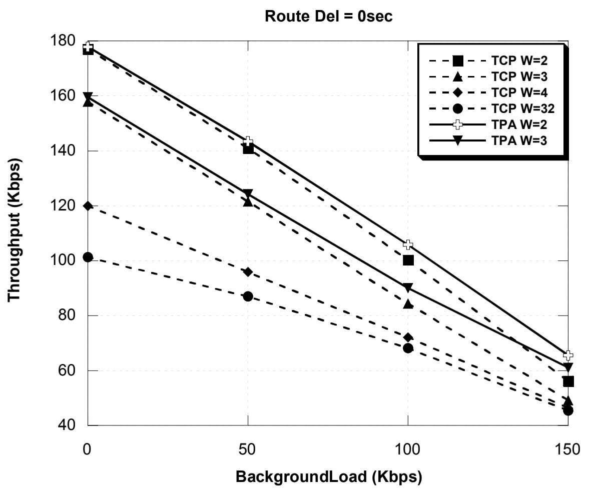
(a) (b)

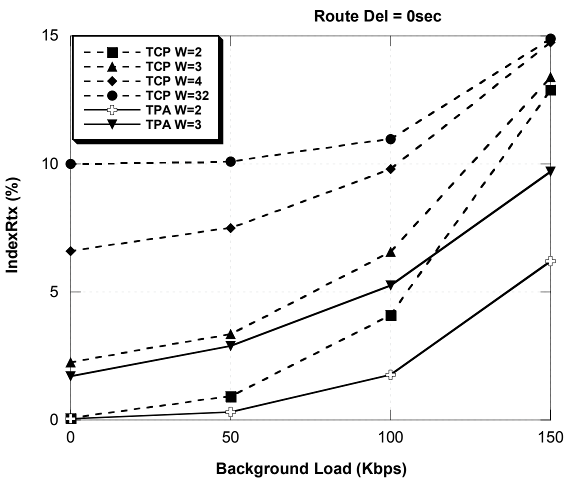
Figure 3.19. Performance of the TCP and TPA protocols with different window size: throughput (a) and retransmission index (b).

In the following sections we will focus on the performance of TPA and TCP with maximum window size equal to 2. However, for completeness, in plots we also show curves related to maximum window sizes different than 2. We will investigate how the performance of both protocols are influenced by factors such as interfering traffic produced by others sessions, latency for discovery a new route whenever a route failure occurs, and presence of a selfish node along the connection path (i.e., an intermediate node that does not cooperate in forwarding packets towards the final destination).

Impact of the Background Traffic

To investigate the effects of the background traffic we ran a set of experiments with the CBR session active. In this set of experiments we assumed route-recovery latency equal to 0, i.e., we assumed that a new route is immediately found after a route failure. The results in Figure 3.20 show that TPA tends to outperform TCP as the background load increases. This is especially evident with respect to the retransmission index. Even with a maximum window size equal to 2, when the background load reaches 150 Kbps, the percentage of retransmitted packets is halved by TPA (6% vs. 12%). In addition, TPA provides a higher throughput. This means that the TPA provides higher throughput with respect to TCP, while – *at the same time* – consuming roughly half of the energy spent by TCP in retransmissions.





(a) (b)

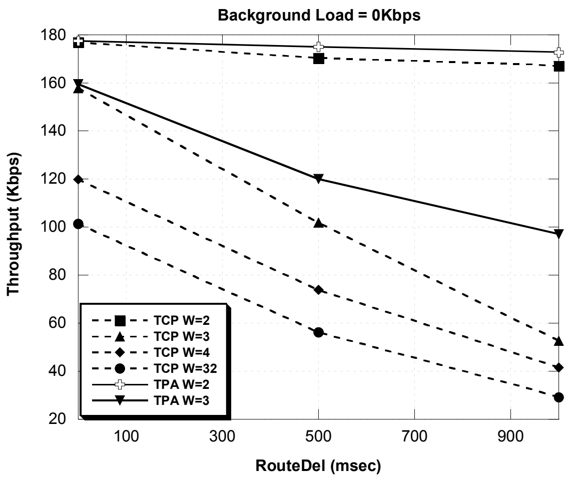
Figure 3.20. Impact of the Background Traffic: throughput (a) and retransmission index (b).

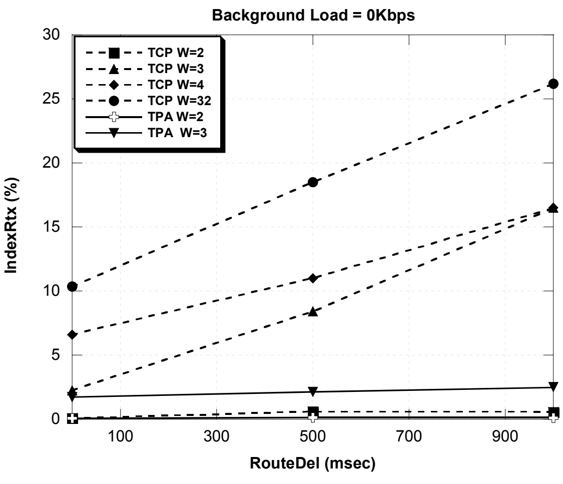
Impact of the Route Discovery Latency

In a static MANET route failures occur due to link-layer contentions. Whenever a route failure is detected, the routing protocol tries to find an alternative route. The discovery of the new route may take some time. In our experiments a fixed delay parameter, RouteDel, represents the amount of time during which no route to the destination is found. Plots in Figure 3.21 and Figure 3.22 show the effects of increasing route discovery latencies. Figure 3.21 shows that, in the absence of background traffic, the route discovery latency has no meaningful effect. TPA only provides small improvements over TCP with window size equal to 2. This means that the policy adopted by TCP and TPA to update the retransmission timer is able to limit the number of retransmissions even for large route discovery latencies (see Section 3.4.2.2).

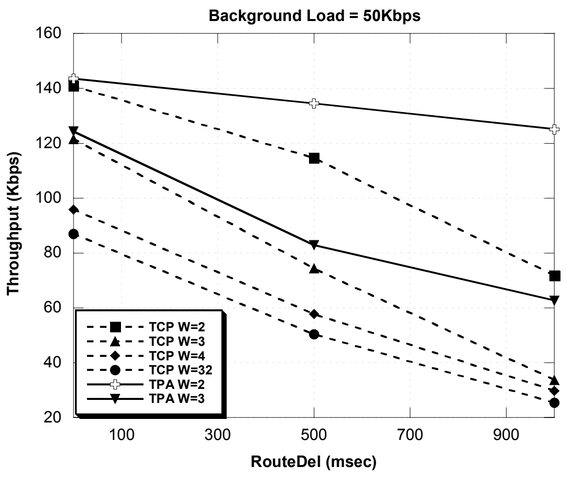
Things change when there is interfering traffic. Figure 3.22 shows that, even with moderate background traffic (i.e., 50 Kbps), TPA largely outperforms TCP both in terms of throughput and percentage of retransmissions. Specifically, when the route-discovery latency is 1 second, the throughput achieved by the TPA is 124.9 Kbps, while the TCP throughput drops to 71.8 Kbps. Furthermore, the TCP retransmission index grows up to 8.9 %, while the TPA retransmission index remains close to 1%. These large performance differences stem from the fact that TPA leverages buffer functionalities that are implemented in typical MANET network protocols. MANET routing protocols (e.g., DSR) usually buffer packets generated by the sender while a new route is being searched. When the congestion level in the MANET increases, the number of packets required to find a new route increases as well. In this case, the TCP protocol keeps retransmitting always the same packet until the new route is found, since its congestion window size is stuck at 1. On the contrary, the TPA protocol transmits successive packets in the main or retransmission streams. These packets are buffered at the sending-node network layer, and thus they are immediately delivered once a new route is found. Ultimately, when the TPA protocol is used, fewer retransmissions occur.

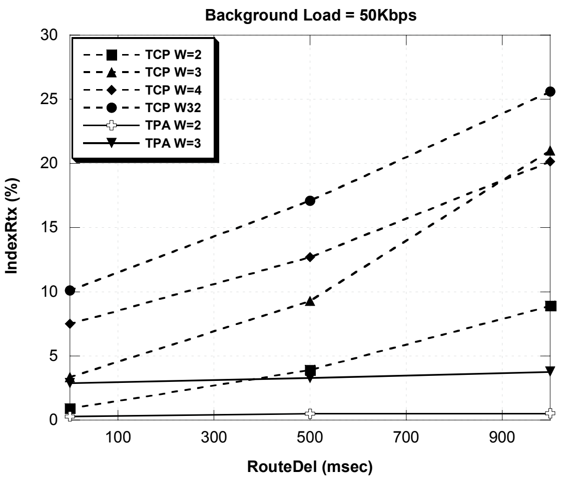
We performed additional experiments with increasing background traffic and found that the difference between TPA and TCP performance becomes larger and larger as the background traffic grows up. Thus, we can conclude that the combined effect of the interfering traffic (causing congestion and, hence, route failures) and large route-discovery latencies has a severe impact on the performance of the TPC protocol. The performance degradation suffered by the TPA protocol is far less pronounced.





(a) (b)
Figure 3.21. Impact of the Route Discovery Latency (without Background Traffic): throughput (a) and retransmission index (b).

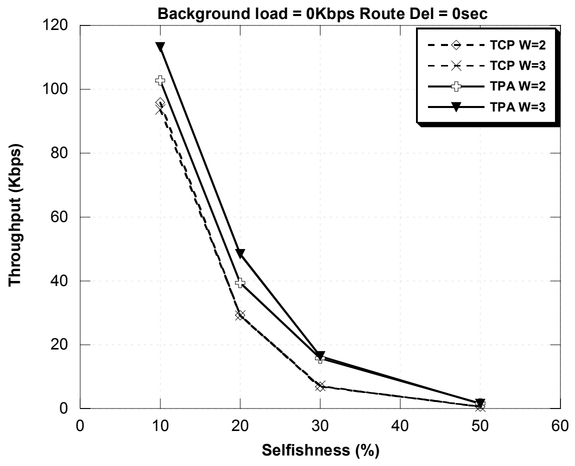


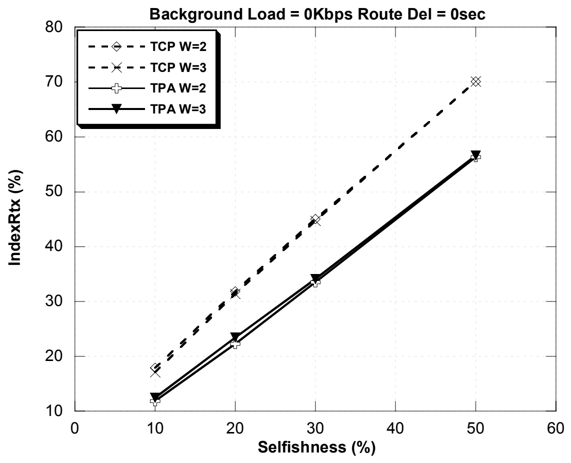


(a) (b)
Figure 3.22. Impact of the Route Discovery Latency (with Background Traffic): throughput (a) and retransmission index (b).

Impact of the node selfishness

In this section we investigate the impact of node selfishness. MANETs relay on the assumption that intermediate nodes are willing to forward data traffic originated by other nodes towards the final destination. However, an intermediate node might not be cooperating either because it is selfish or because it has limited energetic resources (that are deserved to local traffic). In our simulations we modelled the behaviour of a selfish node by assuming that it does not forward (i.e., it discards) a packet generated by another node with probability *p* (*p* defines the degree of node selfishness). Specifically, in our experiment setup (see `Figure 3.18`) we assumed that the selfish node is node 3. Figure 3.23 shows the performance of TPA and TCP for different values of node selfishness. Even in this case, the advantage of using TPA instead of TCP resides both in the lower number of retransmissions, and in the higher throughput. At a selfishness level of 10%, the TPA requires around 34% less retransmissions than the TCP, and the TPA throughput is 7% higher than the TCP throughput. At a selfishness level of 50%, the TPA throughput is 150% higher than the TCP throughput, and the TPA retransmission index is 20% lower. Finally, it is worth noting that plots in Figure 3.23 are derived by setting both the background traffic level and the route recovery latency to 0. In the case of congested networks, the TPA protocol is expected to perform far better than the TCP protocol, both in terms of throughput, and in terms of retransmissions.





(a) (b)

Figure 3.23. Impact of node selfishness: throughput (a) and retransmission index (b).

3.4.4. Summary and Discussion

We have developed a novel transport protocol for ad hoc networks, TPA that is specifically tailored to the characteristics of the MANET environment. This proposal is motivated by the evidence that the TCP protocol exhibits poor performance in a MANET environment. The ultimate reason for this is that MANETs behave in a significantly different way with respect to traditional wired networks, like the Internet, for which the TCP protocol was originally conceived.

The TPA protocol provides a reliable, connection-oriented type of service, and includes several innovations with respect to the legacy TCP protocol. Unlike TCP, TPA is able to manage efficiently route failures and route changes that may arise due to nodes' mobility. In addition, the TPA congestion control mechanism is designed by taking into account the real nature of congestion phenomena in MANETs. Finally, TPA implements a novel retransmission policy to reduce the number of useless retransmissions and, hence, energy consumption.

We have analyzed the TPA protocol by simulation and we have compared the performance of TPA with that of TCP with limited maximum window size in a static scenario (i.e., assuming that nodes are static). This MANET scenario is known to be one of the most favourable to the TCP protocol, even though its performances are still far from ideal. The results obtained show that, even in a static scenario, TPA outperforms TCP in all operating conditions. Specifically, the TPA protocol is able to conserve energy by avoiding many useless retransmissions, while providing at least the same throughput provided by TCP. In addition, the throughput achieved by the TPA protocol is far higher in the case of highly congested scenarios (i.e., when the joint effect of long route discovery delays and background traffic is taken into account). Furthermore, the difference in performance is expected to be even greater in a mobile scenario because, unlike TCP, TPA includes mechanisms to manage efficiently route failures and route changes caused by nodes' movements. The analysis of TPA performance in a mobile environment is left for further study.

3.5. References

[1] E. Belding-Royer, "Routing Approaches in Mobile Ad Hoc Networks", in *Mobile Ad Hoc Networkings,* S. Basagni, M. Conti, S. Giordano, I. Stojmenovic (Editors), IEEE Press and John Wiley and Sons, Inc., New York, 2003.

[2] T. W. Chen, M. Gerla, "Global State Routing: A new Routing Schemes for Ad-Hoc Wireless Networks", *IEEE ICC'98*, June 1998.

[3] G. Pei, M. Gerla, T. W. Chen, “Fisheye State Routing In Mobile Ad Hoc Networks”, *Proceedings of the 2000 ICDCS Workshops*, Taipei, Taiwan, Apr. 2000.

[4] S. Ramanathan, M Streenstrup, “Hierchically-organized, Multihop Mobile Networks for Multimedia Support”, ACM/Baltzer Mobile Networks and Applications, Vol. 3, No. 1, pp 101-119, June 1998.

[5] C. A. Santivanez, B. McDonald, I. Stavrakakis, R. Ramanathan, “On the scalability of Ad Hoc Routing Protocols, *Proceedings of INFOCOM 2002*, New York, June 2002.

[6] C. A. Santivanez, I. Stavrakakis, R. Ramanathan. “Making link-state routing scale for ad hoc networks”. In *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC’01)*, 2001.

[7] R. Koodli and C. E. Perkins, "Service Discovery in On-Demand Ad Hoc Networks", IETF Internet Draft, October 2002.

[8] J. Luciani, G. Armitage, J. Halpern, N. Doraswamy, "Server Cache Synchronization Protocol (SCSP)”. RFC 2334. April 1998.

[9] MobileMAN: Architecture, protocols and services, Deliverable D5, http://cnd.iit.cnr.it/mobileMAN/pub-deliv.html

[10] D.B. Johnson, D.A. Maltz and Y.-C. Hu, “The dynamic source routing protocol for mobile ad hoc networks (DSR)”, Internet Draft of the IETF MANET Working Group, July 2004.

[11] C. Perkins, E. Royer and S. Das, “Ad-hoc on demand distance vector (AODV) routing”, RFC 3561, July 2003.

[12] K. Chandran, S. Raghunathan, S. Venkatesan, R. Prakash, “A Feedback Based Scheme for Improving TCP Performancein Ad-HocWireless Networks”, Proceedings of ICDCS '98, pp. 472-479.

[13] G. Holland and N. Vaidya, “Analysis of TCP performance over mobile ad hoc networks”, Wireless Networks, Vol.8, pp. 275-288, 2002.

[14] J. Liu and S. Singh, “ATCP: TCP for Mobile Ad Hoc Networks”, IEEE J-SAC, Vol. 10, No. 7, July 2001.

[15] D. Sun and H. Man, “ENIC - An Improved Reliable Transport Scheme for Mobile Ad Hoc Networks”, Proceedings of the IEEE Globecom Conference, 2001.

[16] D. Kim, C. Toh and Y. Choi, “TCP-Bus: Improving TCP Performance in Wireless Ad-Hoc Networks”, ICC, 2000.

[17] F. Wang and Y. Zhang, “Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response”, MobiHoc 2002.

[18] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang and M. Gerla, “The Impact of Multihop Wireless Channel on TCP Throughput and Loss”, Proceedings of IEEE INFOCOM 2003, San Francisco (CA), March 30.April 3, 2003.

[19] S. Xu, T. Saadawi, “Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks”, Computer Networks 38 (2002), pp. 531.548.

[20] K. Xu, S. Bae, S. Lee and M.Gerla, “TCP Behavior accross Multihop Wireless Networks and the Wired Internet”, The Fifth International Workshop on Wireless Mobile Multimedia (WoWMoM 2002), Atlanta (GA), September 28, 2002.

[21] P. Gupta and P.R. Kumar, “The Capacity of Wireless Networks”, IEEE Transactions on Information Theory, Vol. 46, No. 2, pp. 388-404, March 2000.

[22] P. Gupta, R. Gray, and P.R. Kumar, “An Experimental Scaling Law for Ad Hoc Networks”, http://black.csl.uiuc.edu/~prkumar/postscript__les.html, 2001.

[23] A.Bakre, B.R.Badrinath, “Implementation and Performance Evaluation of Indirect TCP”, IEEE Transactions on Computers, Vol.46, No.3, March 1997.

[24] W.R. Stevens, “TCP/IP Illustrated”, Vol. 1, Addison Wesley, 1994.

[25] V.D. Park and M.S. Corson, “A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks”, Proceedings of IEEE INFOCOM '97, Kobe, Japan, 1997.

[26] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, “Atp: A reliable transport protocol for ad-hoc networks,” in *Proc. of the 4th ACM Symposium on Mobile Ad Hoc Network and Computing (MobiHoc 2003)*, Annapolis, Maryland, U.S.A., June 2003.

[27] F. Calì, M. Conti and E. Gregori, “IEEE 802.11 wireless LAN: capacity analysis and protocol enhancement”, *IEEE Transactions on Networking*, Vol. 8, No. 6, pp. 785-799, Dec. 2000.

[28] I. Chlamtac, M. Conti, J. Liu, “Mobile Ad hoc Networking: Imperatives and Challenges”, *Ad Hoc Network* Journal, Vol.1 N.1 January-February-March, 2003.

4. COOPERATION MECHANISMS AND MODELS

Ad hoc networks are distributed systems composed of autonomous entities, which need cooperation in order to work properly. In the ad hoc networking technology, mobile nodes equipped with wireless network interfaces freely and dynamically self-organize into temporary network topologies, allowing people and devices to communicate without any pre-existing network infrastructure. The underlying concept is the exploitation of synergy, resulting from collaboration among the network components, to provide services to each other.

The basic requirement for making operational the cooperative paradigm is the supposed contribution of all entities that compose and, at the same time, make use of the system. However, the recent deployment of ad hoc networks for civilian applications, relaxes the assumption on nodes cooperation. As long as applications of mobile ad hoc networks envision mainly emergency and military situations, all nodes in the network belong to a single authority and have a common objective. Therefore, cooperation among nodes can be assumed. In the context of civilian applications, the nodes typically do not belong to a single authority, and consequently nodes cooperation cannot directly be assumed.
The lack of any centralized authority, guaranteeing the overall collaboration, motivates a possible tendency of entities to misbehave by not adhering to the cooperative paradigm.

An entity that does not cooperate is called misbehaving. Cooperation misbehavior can be caused by entities that are malicious or self-interested. A malicious entity aims at breaking the cooperative paradigm to intentionally damage others. This kind of misbehavior gives rise to several security problems, mainly Denial-of-Service (DoS) attacks, ranging from routing/forwarding disruption to resource consumption [HJP02]. An example of a routing disruption attack is for an attacker to send forged routing packets to create a routing loop or to partition the network. An example of a resource-consumption attack is for the attacker to inject extra data packets into the network, which will consume bandwidth resources when forwarded, especially over routing loops.

On the other hand, a self-interested entity does not intend to directly damage the overall functioning, but is unwilling to spend its resources on behalf of others (*node selfishness*). From this kind of misbehavior arises a new class of problems that we group in non-cooperation problems.

This section presents an analysis of the security issues related to *node selfishness*. The impact of such misbehavior is presented from a layered perspective as it entails severe degradation of performance at different levels of the protocol stack. *Cooperation enforcement mechanisms* cope with node selfishness and provide incentives for node to correctly participate to the network operation by making unattractive the natural tendency of nodes to free-ride, i.e. to save their resources for self-interested purposes.

In the sequel of this section -- which is the result of the contribution of four project partners (CNR, Cambridge, Eurecom and SUPSI) -- we first analyze cooperation misbehavior and cooperation enforcement issues by exploiting experiences already performed in computer networking and, more generally, in p2p distributed systems. From this analysis we have been able to derive a generalized model of selfish behavior and pointed out that the basic requirements of cooperation schemes are akin to decision making and economic modeling and a natural tool that emerged as effective in analyzing mobile ad hoc networks in the presence of selfish nodes is game theory. Then we discussed the limitations of cooperation enforcing systems based on distributed algorithms which require nodes cooperation. Hence we introduce our model which is based on a local policy. A node implements its cooperation enforcing policy using only its local information to determine the other nodes reputation. Using this information it implements a local policy for cooperation enforcing.

In the scope of MobileMAN, the ***CORE*** cooperation enforcement mechanism has been selected as the more suitable cooperation scheme for the objectives of the project. The properties offered by CORE are in line with the cross-layering architecture proposed in deliverables D5 and D10; at the same time, CORE has been shown to outperform other cooperation schemes available in the literature when realistic network assumptions are taken into account: for example if we assume that unreliable wireless communications that impact the monitoring component used by CORE and other cooperation enforcement mechanisms, it has been shown in [MM04] that CORE cope well with the *imperfect monitoring* assumption.

As MobileMAN paradigm highly depends on the attitude of humans to cooperate, we felt important to provide a social-science perspective of cooperation in autonomous and self-organized societies. The hard question we try to answer is: "Under what conditions does an individual **voluntarily cooperate** to pursue a common goal, such as sharing his MobileMAN device with other users, as to allow the system to work properly?" Indeed, the viability of ad hoc networks in general and the MobileMAN paradigm in particular heavily depends on users' willingness to cooperate. However, as theory and extensive empirical research shows, cooperation cannot be taken for granted since there are often clear discrepancies between individual goals and collective interests. In this section we provide a review of "*collective action*" theory, which demonstrates that people's willingness to cooperate depends on a number of factors that need to be taken into account in the development of any ICT whose viability is contingent upon cooperation.

4.1. Cooperation issues in Ad Hoc Networks and P2P systems

In this section we focus on self-interested behavior, and explore the literature to investigate the cooperation issues across all layers of the protocol stack for an ad hoc network node. In this context, we refer to the classical TCP/IP MANET architecture grouping the networking activities into link, network, transport, and middleware/application layer.

Ad hoc networking shares many concepts, such as distribution and cooperation, with the peer-to-peer (P2P) computing model [SGF02], which constitutes a natural paradigm for the ad hoc computing model. A defining characteristic of P2P systems is their ability to provide efficient, reliable, and resilient routing between their constituent nodes by forming structured ad hoc topologies on top of a real network infrastructure (see Figure 4.1. Illustration of P2P services built on top of an overlay network abstraction). The difference with traditional distributed computing systems is the lack of a central authority controlling the various components; instead, nodes form a dynamically and self-organizing system. The applications best suited for P2P implementation are those where centralization is not possible, relations are transient, and resources are highly distributed [PC02]. In particular, the range of applications covered by the P2P model includes file sharing, distributed search and indexing, resource storage, collaborative work, etc. The key aspect of P2P systems is the ability to provide inexpensive but at the same time scalable, fault tolerant and robust platforms. For example, P2P sharing systems, like Gnutella [AH00], are distributed system where the contribution of many participants with small amounts of disk space results in a very large database distributed among all participant nodes. The voluntary nature of nodes' contribution has also serious drawbacks as the system's resources can be highly variable and unpredictable. Furthermore, the lack of a central authority coordinating the resources that each peer should contribute, leads users to use the system without contributing much to it. The non-cooperation problem, *free-riding* in P2P terminology, highly affects the system performance, leading it to turn its peer-to-peer spirit in a more traditional client-server one [SGG02].

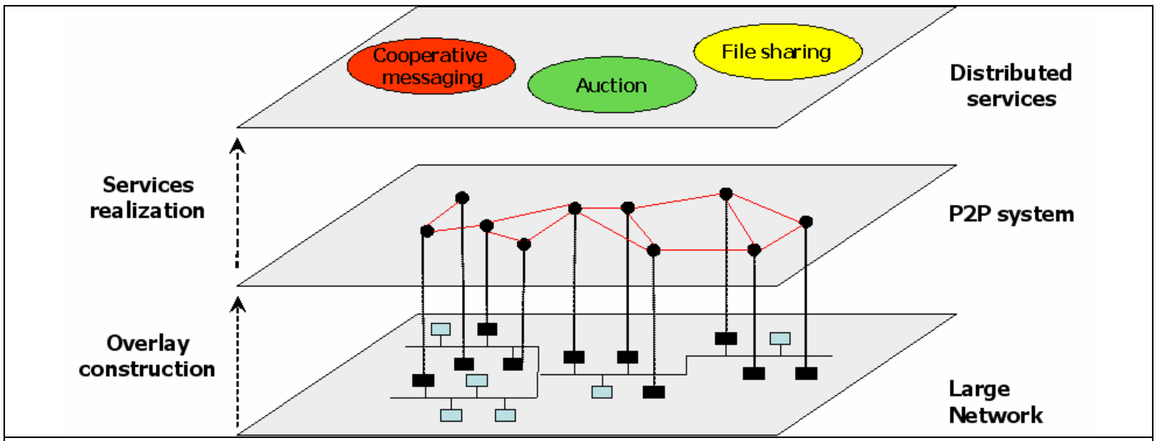


Figure 4.1. Illustration of P2P services built on top of an overlay network abstraction

The P2P paradigm not only fits well the requirements to develop applications for the ad hoc context, but it can also be considered a test-bed for studying cooperation issues. While only small prototypes of ad hoc networks exist, large P2P systems are currently operational, and from these systems we can gain insight on the attitude of users to cooperate. For this reason, we firstly consider P2P cooperation issues, and then we draw useful parallels with the ad hoc wireless networking.

4.1.1. Lessons from P2P

Nowadays, selfish behavior represents the main hot topic in the cooperation research area of ad hoc networks, in contrast with malicious behavior which is the basis of the security area [MM03]. However, the cooperative paradigm adopted by the ad hoc networking technology is not new to the research community, and notions of *rationality* and *self-interest* have been already studied in the P2P context [SP03].

The P2P computing model assumes that users will follow prescribed protocols without deviation. For example, in a search protocol like Gnutella, the participating peers constitute an overlay network, where each peer is connected to a number of other peers, or neighbors. As shown in Figure 4.2. Gnutella, an example of fully decentralized P2P system, when a user submits a query, the corresponding peer will send the query message to all its neighbors, which will in turn forward the query to their neighbors, and so on. Peers that can answer to the received query will send a response to the querying peer. The querying peer, after waiting a period of time for responses to arrive, will select one or more responding peers from which to buy the service. Clearly, the search mechanism relies on cooperation among peers to forward queries.

The assumption on peers' cooperation ignores the user's ability to deviate its behavior for self-interested reasons. This is because nodes represent users who can modify their behavior by considering what is best for them (rational users). Behavior deviations depend on the type of system: free or remunerative. In the context of free systems, the *free-riding* problem has become a central issue. Free-riders consume a resource but do not produce at the same level of their consumptions. For example, in a free file-sharing, peers acting in their own best interest conserve their resources (i.e. bandwidth) by sharing no files. Hence, only a small fraction of altruistic users offer almost all the available content. A measurement conducted by sampling messages on the Gnutella network over a 24-hour period [AH00] has established that almost 70% of Gnutella users share no files, and nearly 50% of all responses are returned by the top 1% of sharing hosts. In the worst case scenario, the ``tragedy of commons'' [H68] may be inevitable: the overall bad functioning is not created by malicious external agents, but by the apparently appropriate and innocent behavior of many individuals (peers) acting alone. Although the degradation due to each self-interested entity is small, if all entities follow this pattern the commons will ultimately be destroyed.

On the other hand, in applications where peers gain from answering queries (e.g. pay-per-transaction file-sharing systems, P2P auctions, and P2P service discovery systems), the opposite extreme of self-interest comes up: peers become eager to provide service, and hence they are in *competition* with others to provide their services. The competition problem mainly regards P2P frameworks that rely on peers to forward queries: a peer acting in its own best interest will not forward queries to potential competitors. For example, a peer providing a car rental service might not forward a query for car rental services, issued by a peer looking for that service. Instead, it could answer the query and then drop it, so as to improve its chances of gaining business. Another example concerns distributed auctions [SP03]. An auctioneer sends out an announcement advertising its service and asks its neighbors to globally propagate it. It then expects to receive several bids. Surprisingly, it receives only as much bids as its neighbors. This is because its neighbors understood that it was in their best interest not to forward the initial announcement in order to limit the competition and hence win the auction.

To summarize, cooperation misbehavior in P2P systems depend on the economic nature of the system: free or remunerative. While free systems motivates nodes toward *selfishness* (free-riders are simply consumer, and do not contribute much to the system), remunerative systems leads to the other extreme of self-interest, *greediness* (greedy entities are in competition to increase their profit).

Most of research on cooperation enforcing in P2P systems presents solutions based on incentives. The basic idea is very simple: users must be encouraged to balance what they take from the system with what they contribute. A natural approach is to charge users for every service use, and to reward them for every service provision.
This is done by means of micro-economics schemes. They mainly deal with the investigation of different policies to apply micro-payment mechanisms. In [GLML01] three possible variants of a micro-payment scheme are presented for a file sharing system. In the first one, a user executes a micro-payment every time he downloads a file, while receives a micro-payment every time he uploads a file. A second variant of this scheme, called quantized micro-payment scheme, introduces a mechanism where users pay for downloads in block of *b* file, where *b* is a fixed number, while the pricing mechanism for serving files is unchanged. The third approach proposes rewards for sharing: users continue to pay for downloads, while rewards depend on the amount of material they share, rather than the number of uploads they provide.

In [KYG03], an economic protocol to incentive cooperation in the face of competition is proposed. Incentives are represented by the concept of a ``right to respond'', or RTR, that is a token indicating that a peer has right to respond to a query message. The protocol aims at stimulating peers to forward queries, by making them get paid when doing it. Once a peer has a

RTR for a given query, it may respond to the query and/or sell the RTR to other peers. If a peer sells a RTR, it may still respond to the query corresponding to the RTR. That is, a peer does not lose the right to respond to a query when it sells the RTR for that query. Peers can buy and sell RTRs with their neighbors only. However, this model introduces a double form of payment (RTR and money) whose realization may result complicated.

A solution to enforce fair sharing of peer-to-peer resources is presented in [NW03]. The distinguishing feature is the design of a natural incentive directly into the P2P system. Fair sharing is supported by limiting any given node to only consuming as much of the network's resources as it provides space for others on its local disk. In this system there is no need for cash or other forms of money as incentives are in the form of a barter economy. When nodes exchange their local storage for others' remote storage, the trade benefits both parties, giving an incentive for them to cooperate. To accomplish the resource provision/usage balancing, each node is required to maintain a ``usage file'' containing information on its local and remote storage. Locally storing is on behalf of others nodes, while the node benefits of remote storages. These records allow understanding if a node is under quota, i.e. the difference between its local advertised capacity and the amount of space remotely occupied is positive. An under quota node is allowed to write new files into the network, if the new storage does not brings it over quota. A mechanism of random auditing among nodes discourages nodes from cheating (e.g. a node might inflate its advertised capacity or deflate the sum of its remotely stored files).

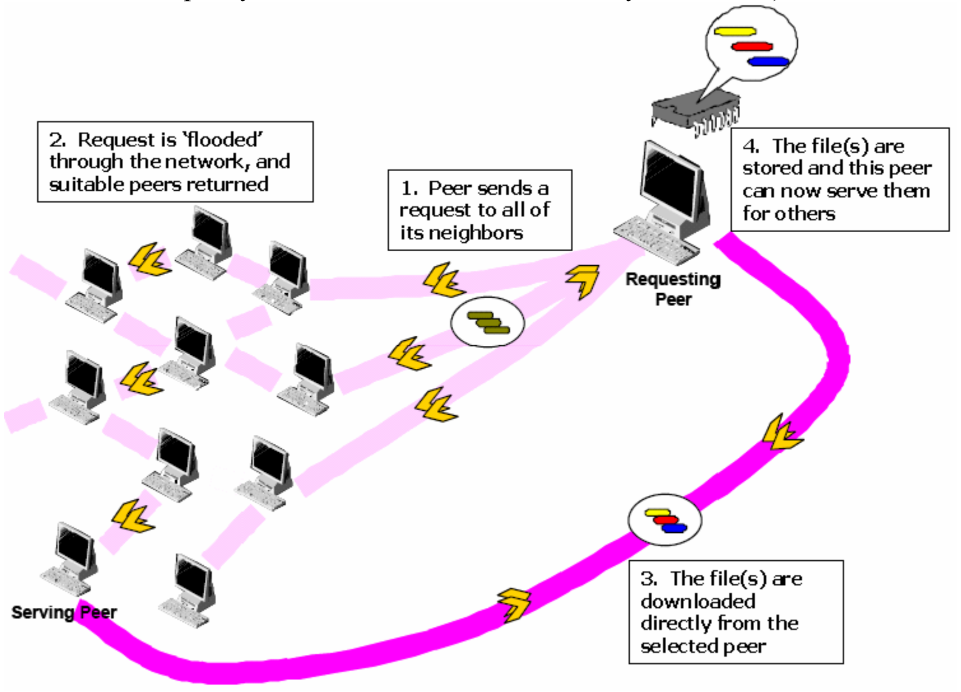

2. Request is 'flooded' through the network, and suitable peers returned
1. Peer sends a request to all of its neighbors
4. The file(s) are stored and this peer can now serve them for others
Requesting Peer
3. The file(s) are downloaded directly from the selected peer
Serving Peer

Figure 4.2. Gnutella, an example of fully decentralized P2P system

These schemes have proved effective and may inspire solutions for the ad hoc context. For example, the fully distributed nature of the last proposed scheme seems suitable for application in ad hoc environments. Random auditing among ad hoc nodes could support misbehavior detection. Credit based systems may be useful as well, but some open issues must be considered: micro-payments involve the employment of some electronic currency that needs some sort of security mechanisms. In the ad hoc perspective, some additional questions arise. Architectures with centralized authorities cannot be directly assumed in the ad hoc context and the applicability of micro-payment mechanisms is still an open problem. Even in the absence of cash or other forms of money as incentives, like in the last presented solution, an evaluation of produced overhead is needed to estimate effectiveness in the ad hoc context.

4.1.2. Selfishness in Ad Hoc Networks

In emerging applications for ad hoc networks, the nodes typically do not belong to a single authority, and hence cooperative behavior cannot directly be assumed. Providing service to each other consumes resources, which are generally scarce on ad hoc nodes. Furthermore, nodes are supposed to be *rational*, in the sense that they try to maximize their own utilities in a self-interested way by not adhering to the protocols specification.

Selfishness is a new problem for the ad hoc environment, and characterizes a class of intentional but not malicious misbehavior. In particular, a node may act selfishly whenever it is required to forward packets on behalf of others, without being directly rewarded. Far off nodes communicate using intermediate nodes as relays. Depending on their position, nodes may be often required to forward packets for the benefit of others, consuming a lot of energy and thus leading to new possible selfish behavior.

The presence of selfish nodes may significantly degrade the performance of the ad hoc system. Network services may not be available and cooperative nodes may become overloaded, possibly leading to new cooperation misbehavior. In the worst case scenario, the network may become partitioned.

In the following, we go through the TCP/IP protocol stack for ad hoc nodes in order to present cooperative aspects of each protocol. Each layer is characterized by cooperation issues due to selfishness. In particular we present selfish behavior defined in literature for the MAC, network, and transport layers. Research on cooperation at the middleware/application layer is still in its infancy and hence we omit the middleware/application section. However, issues regarding this layer recall aspects already described in the P2P environment.

MAC-layer misbehavior

All nodes within a wireless ad hoc network use the same frequency band as a shared medium for receiving and transmitting data. MAC protocols such as IEEE 802.11 [IEEE99] aim at controlling channel contention to prevent unfair channel sharing. The IEEE 802.11 MAC protocol is based on a fully distributed mechanism, called Distributed Coordination Function (DCF), for resolving contention among multiple nodes accessing the channel. A node with a packet to transmit initializes a back-off counter with a random value selected from a range [0, *CW*], where *CW* is a variable maintained by each node called the *Contention Window*. While the channel is idle, the back-off counter is decremented by one at each time slot. The counter is frozen when the channel becomes busy. The node may transmit a packet when the back-off counter reaches zero. If the transmission is successful, *CW* is reset to an initial value CW_{min}. If the transmission is not successful (i.e. because of a collision with another node's transmission), *CW* is doubled, up to a maximum of CW_{max}.

All the participating nodes are required to adhere to the back-off protocol to ensure a fair share of bandwidth for each node in the long run.
A selfish node may fail to adhere to the contention resolution protocol, with the intent of obtaining more than its fair share of the channel bandwidth [KV03], by

- selecting smaller back-off values, different than the distribution specified by the protocol (e.g. by selecting back-off values from a smaller range than [0,CW]);

- using a different retransmission strategy not doubling the Contention Window value after a collision.

The selfish aspect shows up in the node unwillingness to wait for a fair time before transmitting.

Network-layer misbehavior

In MANETs, basic network functions like routing and forwarding are distributed over all the participating nodes [RT99]. Every node must act as a router, and far off nodes communicate using intermediate nodes as relays. However, these functions may be really critical under a resources consumption point of view. Nodes may be often required to forward packets for the benefit of others, thus spending big quantities of energy. Selfishness at the network layer mainly regards:

1. Forwarding: The node does not perform the packet forwarding function. Each packet that has a source or destination address different from the current node is discarded.
2. Routing: The node does not perform the routing function. It discards every routing packet not of its interest, without relaying the packet to its neighbors. There will be no route including that node, and hence it will never be asked to forward a packet on behalf of others.

Selfish behavior may depend on nodes energy level. Hence, a node can behave differently according to the available energy: it can switch between the two types of selfishness (routing or forwarding), or it can assume both at the same time. A selfishness model for the routing/forwarding functions is defined in [MM02], where authors refer to the DSR protocol and describe three types of selfish behavior, depending on node's energy level. In case of good energy level the node behaves properly, executing both the packet forwarding and the routing function. When the energy level falls down a first threshold, the node stops executing the forwarding function, but still cooperates to routing. When the energy level decreases and falls down a lower threshold, it disables also the routing function.

A simulation study on the impact of this model on the DSR protocol has shown that selfishness can cause serious damage in terms of global network throughput and delay [MM02].

Transport-layer

At the transport layer, cooperation issues mainly regard misbehavior already identified for the TCP congestion control mechanism in wired networks [SCWA99]. If the sending endpoint misbehaves, and does not obey the appropriate congestion control algorithm, then it may send data more quickly than well-behaved hosts, forcing competing traffic to be delayed or discarded. On the other hand, a misbehaving receiver can achieve an analogous result to receive data more quickly. In particular, data transmission rate can be increased to an arbitrary large value by affecting the correct operation of TCP congestion control at the sender, by means of *Ack division*: upon receiving a data packet containing N bytes, the receiver divides the resulting acknowledgment into M, where $M =< N$, separate acknowledgments, each covering one of M distinct pieces of the received data segment.

This leads the TCP sender to grow the congestion window at a rate that is M times faster than usual. The receiver can control this rate of growth by dividing the segment at arbitrary points, up to one acknowledgment per byte received (when M = N).

Besides misbehavior inherited from the wired context, research on cooperation issues at this layer has not identified problems typical of the ad hoc context. Instead, most of efforts are oriented to improve TCP performance. In the following, we identify the aspects of current research that can be subject to cooperation misbehavior.

So far, research has explored new approaches to improve TCP performance over wireless links. In ad hoc networks, events such as route failures and route changes due to nodes' mobility can cause serious problems to TCP. Route failures can cause packet drops at intermediate nodes, which will be wrongly interpreted as congestion problems. Route changes can introduce frequent out-of-order delivery, which will further confuse the TCP control mechanism. Current solutions on performance improving [CRVP01] [HV99] [LS01] depend on notification or feedback from the network or a lower layer. They vary in how to obtain feedback and how to respond accordingly. Such mechanisms rely on cooperation of intermediate nodes that are requested to notify the sender, in different ways, about route failures, so that it stops sending packets until it is notified of the restoration of the route. This avoids the sender to respond to the failures as if congestion happened. On the other side, such mechanisms introduce new possibility for nodes to misbehave. In fact, intermediate nodes do not get any advantage from cooperating to such TCP-performance improving mechanisms, and may act selfishly by not sending route failure notifications to the sender. Such a selfish behavior makes the TCP-performance improving mechanism useless because the sender node will understand the lack of acknowledgment as a congestion event.

4.1.3. A new taxonomy of Self-interested Behavior

The study of P2P issues highlights the two opposite aspects of self-interest (selfishness and greediness), and their dependence on the economic nature of the realized system. Nodes tend to be selfish whenever they do not gain anything from providing a service, and thus they aim at saving resources for their own activity. On the other hand, in remunerative systems nodes get eager to provide service, and hence they are in competition with others. Greediness characterizes remunerative service provision, as nodes try to maximize their profit.
In ad hoc networks, only selfish behavior has been so far defined. However, if we translate the economic aspect of P2P self-interested behavior into a more abstract remunerative aspect, we can also identify greedy behavior. In particular, selfish nodes tend to work less, while greedy nodes try to obtain more resources.
According to this point of view we can give a new taxonomy of cooperation misbehavior for ad hoc networks, identifying with:

- *Selfishness*: a node's attitude toward unwillingness to spend battery life, CPU cycles, or available network bandwidth for the common good, without being rewarded. It uses the network services but does not cooperate, even though it expects others to cooperate.
- *Greediness*: the attitude of a node to act greedily, trying to achieve as much as possible for its own interest, in terms of network resources or economic profit. The intent is not of directly damaging others, but of maximizing its own reward, even though it results in an unfair resource/profit share.

The common aspect of these two extremes of self-interest is utilities' maximization, which can be achieved by saving resources, as well as by increasing economic/resources profit.

In order to classify self-interested behavior for ad hoc networks in greedy and selfish behavior, it is necessary to identify the protocol goal: while selfishness characterizes service provision, greediness is motivated by resource sharing.

Let us apply these two models to the protocol stack. Greediness characterizes the link and transport layers as they rule the access to a shared resource. The MAC protocol (link layer) controls the access to the wireless medium. As nodes contend with each other for accessing the channel, they may become greedy to obtain a more than fair channel reservation. Hence, we redefine as greedy, the misbehavior regarding the unfair channel sharing previously described.

The transport protocol is an end-to-end protocol that provides a mechanism for a fair bandwidth share among all nodes. As already identified in wired networks, protocol endpoints may try to achieve more bandwidth than other nodes, thus becoming greedy.

On the other side, selfishness characterizes the network layer, where nodes cooperate to provide a communication service without being rewarded. Hence, nodes may act selfishly by not supporting communications among other nodes.

This layered analysis highlights how only function-dependent models have been defined for self-interested behavior. It is also important to notice that selfishness models defined in literature (e.g. routing/forwarding selfishness depending on nodes' energy level) implies a sort of maliciousness of the user: he must implement a protocol modification, or change some implementation's parameters, in order to discard packets. The result is a behavior not consistent with the protocol specification (packets are not forwarded). Thus, we consider this kind of selfishness as a *combined* selfish-malicious behavior, which can effectively be adopted only by expert users (small percentage). This feature led us to notice that a more general selfishness model is possible. Thus we define

- *Pure selfishness*: a behavior concerning two possible states: ``active" and ``stand-by". As long as the node needs to communicate with the others, it is active and cooperates to the network functions. When it does not need anymore network resources, it goes in a stand-by state, becoming unavailable to other nodes.

Pure selfishness is more general because it regards all network functions at the same time, and it is applicable by every user just turning off its connection to the network. When a node goes in a stand-by state it is disconnected from the network and automatically disables all the network functions (routing, forwarding, etc.) at the same time. This model involves two legal states, active and stand-by, that every user can adopt without any hardware/software modifications.

This new type of selfishness can be further extended in order to include also function-dependent behavior by introducing a *pure restricted selfishness* model. We call this model restricted as it can be applied by ``advanced" users that have the expertise to modify the parameter setting of the protocol stack to avoid supporting packet forwarding and/or routing. This indeed requires only modifying a parameter setting still using the legacy protocol and hence appears as legal behavior. Pure restricted selfishness will comprehend three possible states:

- *All active*
  The node has good energy level and cooperates to all network functions.

- *One/more network function/s*
  The node disables one or more of network functions or services, such as routing and/or forwarding to save resources for its own functioning. The node may also disable middleware services such as free file sharing.

- *All off*
  The node adopts a pure selfish behavior. It disables all network functions by turning off its network connection.

As shown in Table 1, pure selfishness is really easy to adopt (i.e. shutting down the network connection is the most natural selfish behavior). However, as we will show later in the paper, none of the current cooperation enforcing systems addresses this kind of behavior.

Combined selfishness disabling the routing and/or forwarding functions, requires some technical expertise. Operating systems such as Linux allow disabling the routing/forwarding function through a command, called `iptables`. Complexity in applying combined selfishness increases when the behavior is dependent on the node's energy level: the user must be able to dynamically disable routing and forwarding according to the device's energy. Most of work on cooperation enforcing solutions deals with combined selfishness. In the following we go through current approaches.

Selfishness type	Application complexity	Proposed solutions
Pure	None	none
Combined: No routing/forwarding	easy (command)	network-layer solutions
Combined: energy level dependent	Medium (program)	Network-layer solutions

Table 1. Selfishness classification

4.1.4. Cooperation enforcing solutions: a layered analysis

Research on cooperation enforcing mechanisms is proceeding separately for each layer of the protocol stack. The MAC and transport layers present few solutions, addressing cooperation issues not typical of ad hoc networks: MAC misbehavior is inherited from wireless networks, while transport issues come from the wired context. The network layer presents the most prominent issues, which have arisen with the ad hoc context. At the application level cooperation issues have not yet been faced but they recall concepts faced by P2P systems.

Current solutions share some features, and can be generally grouped according to the mechanism adopted to enforce cooperation:

1. modifying the protocol to incorporate some features that avoid misbehavior;
2. providing a mechanism based on incentives to reward well-behaving nodes and to punish misbehaving ones.
   Incentives can be of two types:
   - reputation-based (good reputation can be seen as an incentive)
   - price-based

Protocol modifications regard the MAC and transport layers, where changes respectively to the back-off calculation and congestion control are proposed. At network layer we mainly find solutions adopting incentive based mechanisms (both reputation and price based). In the following we give a detailed description of proposed solutions for each layer.

MAC-layer

The problem of a greedy sender has been faced in the context of infra-structured wireless network, with a possible extension to ad hoc networks, based on a reciprocal monitoring of sender and receiver [KV03]. The proposed solution requires some modifications to IEEE 802.11 DCF, and supposes the presence of trusted base stations that can identify sender misbehavior. As base stations are trusted, they are supposed to well-behave (work correctly) while sending data. The problem of a greedy sender arises when a mobile node is communicating with a base station (in infra-structured wireless networks communication between mobiles occurs only through base stations). The proposed solution entrusts the receiver with the task of calculating the back-off value. Instead of the sender selecting random back-off values to initialize the back-off counter, the receiver selects a random back-off value and sends it in the CTS and ACK packets to the sender. The sender is requested to use this assigned back-off value in the next transmission to the receiver. In this way, a receiver can observe the number of idle slots between consecutive transmissions from the sender, and identify eventual senders waiting for less than the assigned back-off. When the receiver perceives a sender deviating from the protocol, it adds a penalty to the next back-off assigned to that sender. In an ad hoc context, where all nodes are un-trusted, the receiver may misbehave in assigning small back-off values to a sender, with the intent of obtaining data from the sender to a higher rate. The approach proposed to handle this misbehavior is similar to that used for detecting sender misbehavior: the receiver can be required to select the initial back-off values using some well-known deterministic function, which the sender is aware of. Hence, the sender can detect a receiver sending small back-off values, and choose to wait for longer interval between transmissions when such misbehavior is detected.

However, the link layer is the most difficult to manipulate, as the MAC protocol is implemented directly on the hardware device. This characteristic provides an indirect form of protection against protocol modifications: changing the on-board firmware requires special equipments and knowledge.

Network-layer

Current approaches to cooperation enforcing at network layer can be classified in two categories: *pricing* based schemes and *reputation* based schemes.

Pricing based schemes stimulate packet forwarding by means of virtual currency (credit). The key idea is that nodes providing a service should be remunerated, while nodes receiving a service should be charged. Based on this concept, [BH03] proposes a tamper-resistant security module which maintains a *nuglet counter*. The proposed protocol requires the node to pass each packet (generated as well as received for forwarding) to the nuglet counter which is decreased when the node wants to send a packet as originator, and increased when the node forward a packet. Hence, if a node wants to send its own packets, it must forward packets for the benefit of others. Besides the drawback of requiring a tamper resistant security module, this solution does not face the problem of nodes running out of nuglets, because of their position on the network borders. The random waypoint model, adopted for nodes mobility during simulations, prevents nodes to be permanently confined at the border of the network (leaf nodes). This avoids the problem of nodes running out of nuglets. In a more static network topology, leaf nodes will rarely be required to forward packets on behalf of others, and hence will not be able to gain credits for their transmissions, consequently running out of nuglets even if they cooperate to the network functioning.

A similar solution, called SPRITE [ZCY03], proposes to overcome the requirement of a tamper-proof hardware by inserting a Credit Clearance Service (CCS) to handle credit. When a node receives a message, the node keeps a receipt of the message. Later, when the node has a fast connection to a CCS, it reports to the CCS that messages that it has received/forwarded by uploading its receipts. The CCS then determines the charge and credit to each node involved in the transmission of a message, depending on the reported receipts of a message. Unfortunately, the centralization of credit handling introduces new issues: i) the policy to choose the node in the ad hoc network that should run the CCS; ii) trustworthiness of such a node. Furthermore, the proposed architecture presents the CCS as a fixed point, reachable by mobile nodes through the

Internet. This constitutes a big constraint for the application of such a system to real ad hoc networks.

The issue of how prices can be determined is addressed in [CG03], where authors consider how incentives can be integrated into the operation of a mobile ad hoc network, so that the cost of resources consumed at transit nodes, when forwarding traffic along multi-hop routes, can be recovered using pricing mechanisms. These prices are determined by individual users according to their bandwidth and power usage, and routes for connections from a user to a particular destination are chosen such that the route price is minimal.

Reputation based schemes discourage misbehavior by estimating nodes reputation and punishing nodes with bad behavior. The starting step for this class of schemes is given by [MGLB00]. They present a solution aimed at detecting and avoiding misbehaving nodes through a mechanism based on a watchdog and a reputation system. The watchdog identifies misbehaving nodes by performing a neighborhood monitoring: it observes the behavior of neighbors by promiscuously listening to communications of nodes in the same transmission range. According to collected information, the reputation system maintains a value for each observed node that represents a reputation of its behavior. The reputation mechanism allows avoiding sending packets through misbehaving nodes. In this way, malicious nodes are rewarded and strengthened, while cooperation enforcing is totally absent.

The following works, CONFIDANT [BL02] and CORE [MM02s], extend such a scheme with a punishment mechanism that isolates misbehaving nodes by not serving their requests. When a neighbor's reputation falls down a predefined threshold, service provision to the misbehaving node is interrupted (this strategy is an approximation of the Tit for Tat). In such a way, there is no advantage for a node to misbehave because any resource utilization will be forbidden. As shown in Section 4.2, CONFIDANT may have severe drawbacks in term of traffic overhead and wrong accusation spreading. The CONFIDANT protocol generates some additional traffic for reputation propagation. The produced overhead may result heavy, and malicious nodes may perform a new attack by sending false alarms about other nodes.

In addition, both protocols may suffer from the watchdog's weaknesses: in presence of collisions, not-homogeneous transmission ranges, or directional antennas, the watchdog is not able to properly monitoring the neighborhood, and misbehaving nodes detection can fail. However, in Section 4.4, by exploiting a non cooperative game model, we show that CORE mechanism is robust with respect to watchdog's weaknesses.

Transport-layer

TCP research on ad hoc networks mainly copes with performance issues due to nodes' mobility. As we previously saw, current solutions for improving TCP performance on ad hoc networks may introduce new cooperation misbehavior from intermediate nodes. Such an issue has not still been faced in research.

Regard misbehavior for the TCP congestion control mechanism already identified in infra-structured network, the solution given in [SCWA99] is feasible also for ad hoc networks. It proposes a couple of modifications to the congestion control mechanism to operate at byte granularity. The first solution suggests incrementing the congestion window proportionally to the amount of data acknowledged, instead of a full Sender Maximum Segment Size (SMSS).

Alternatively, the congestion window can be incremented by one SMSS only when an ACK covering the entire segment sent arrives.

Application/middleware-layer

With the rapid advances in ad hoc networking research, ad hoc networks have attracted a growing interest in developing new applications in the context of personal area networking, home networking, law enforcement operation, search-and-rescue operations, commercial and educational environment. These applications have to face the limitations typical of ad hoc networks and hence will surely share many concepts and issues with peer-to-peer systems. However, cooperation issues at this layer have not yet been considered.

4.2. Arguments against visible incentive systems for ad hoc routing

When all the nodes of an ad hoc network belong to a single authority, e.g. a military unit or a rescue team, they have a common goal and are thus naturally motivated to cooperate. However, for general applications with large numbers of unrelated users, if battery power, bandwidth, CPU cycles and other resources are scarce, selfish users might not wish to forward packets for other users as it would impact their own ability to transmit traffic.
These concerns have resulted in a number of efforts to design incentive systems for mobile ad hoc networks that encourage users to cooperate, as well as trust management systems that identify non-cooperating nodes and punish them. However, these incentive systems have a number of inherent flaws that make them difficult and undesirable to implement in practice. Ironically, if badly implemented, some of them even have the potential to backfire by offering an incentive to cheat the incentives system in order to gain further benefits.

4.2.1. Token based incentive systems

Quality of Service problems

With token-based incentive systems, the basic idea is to use notional credit, monetary or otherwise to pay off users for the congestion costs (transmission and battery costs) they incur from forwarding packets from other users. These credits can then be used to forward their own packets through other users, resulting in an incentive to act as relay points, especially where there is the greatest excess demand for traffic since they earn the most. Users who do not cooperate will not be able to use the network themselves, having not earned any credits.
This idea makes a lot of sense in theory, but when practically implemented is likely to run into a number of problems.
Under the general token mechanism, a user’s token count is increased when it forwards, and decreased proportionally to the number of hops it needs when it sends. This inevitably means that a user needs to forward more than he sends and also limits the amount of information that any user can send at any given time, dependent on their store of tokens. In principle the node may be able to buffer packets until it earns enough to send, but this works only as long as the buffer is large enough and there are no delay constraints on the packets, which rules out many real time applications. Therefore, practically speaking, packets could often be dropped, rendering it somewhat ineffective and inefficient for many types of communications.
Another concern is that a significant amount of energy is thus wasted in the system transmitting dropped packets that would not have been dropped had the incentives scheme not been in place. Because of the wasted energy, a user might find that his battery drained faster than if he were to

cooperate with no incentives system in place, as in both cases he would be forwarding packets for others but with the incentives system he suffers additional energy loss from dropped packets.
The system also puts users on the outskirts of a network at a disadvantage unrelated to their willingness to participate. Those users will not have as much traffic routed through them due to their location and furthermore will have lower congestion prices because of that. They will thus earn significantly less than a centralized node and be penalized for it resulting in low QoS. The system might indeed stabilize overall, but not at a point that is beneficial to everyone.
To pay out credit to forwarding nodes, the transmitting node must estimate the number of hops required so that it can load sufficient credit onto its packet to pay each of the nodes. This calculation not only takes up resources but if done incorrectly will result in packets that have insufficient credit being dropped, as well as wasted credit, decreasing QoS for all concerned.
From a general consumer's point of view, these problems collectively result in dropped packets, excessive consumption of resources and generally poor quality of service for no apparent reason, representing a rather significant drawback to the use of ad hoc devices. Users with poor quality of service are unlikely to be sympathetic (or even aware) to arguments that the system works in such a way for the greater good. This would cause problems not only for individual users, but also for the overall network as unsatisfied users leave the system completely and bad word of mouth discourages new users to join. Ad hoc networks need a critical mass of users to function well, with the utility of the network increasing proportionally to the square of the number of nodes, as stated by Metcalfe's Law.

Technical conundrums

When using tokens, there is also the question of how the balance of tokens can be maintained for users. The average token level within the system needs to be kept at a reasonable level in order for incentives to work properly. If it grows too high, everyone will be rich in tokens and no longer have an incentive to cooperate, and conversely, if there is not enough credit within the system then hardly anyone will be able to transmit. However, if an individual's token level is regularly reset (as proposed in current systems) in order to maintain a certain token level, then there is no incentive to cooperate in the long term. Nodes are free to stop cooperating once enough credit is earned to complete their transmission, since excess credit will be lost anyway.
Some systems propose using real money as credit, either directly or indirectly (to buy virtual credit). In an incentives system this could prove very dangerous, because it would in itself be a strong incentive for users to game the system in order to derive monetary gains. Unless a perfect cheat proof system can be designed, which is rather unlikely, such an incentives system would ironically make it more worthwhile for users to attempt to cheat. The need to pay to communicate would also negate one of the key advantages of ad hoc networks and make it less appealing with respect to competing technologies. Also, any system that involves real money and does not incorporate tamper proof hardware requires a centralized authority. This would undermine the self-organizing, decentralized nature of ad hoc networks, as well as require suitable infrastructure to be built, making the networks less easily deployable and less scalable. It would also be difficult in an ad hoc network to ensure that centralized authorities would always be within coverage.
Tamper proof hardware in turn is very difficult to achieve as suggested in [WP01]; virtually any system can be modified. A determined hacker would be able to compromise a system regardless of whether there was a 'tamper proof' module in place (even if the module was truly tamper proof the hacker might simply replace it with one of his own design). In the end this might only discourage less technically capable users who would not have tampered with the devices in the first place.
Another problem with such systems is that it is very difficult to charge users fairly, without introducing additional complexity. In most systems presented to date it is the sender that always pays, although it is technically possible to also charge either just the destination or both. This is

mainly to prevent the sender from sending useless messages and flooding the network. However, in many cases it is the destination that stands to benefit from a transmission and charging only the sender may thus lead to inconveniences to the user in practice and thereby discourage use of the system. In the same vein, charging just the destination or even both parties would not be perfect solutions either, as the beneficiary changes with each application. (An alternative method of preventing useless messages from being sent might simply be a hardwired mechanism that throttles communications exceeding a certain rate/amount). It is also unclear how this payment issue scales to two-way communications, especially when one side has enough credit and the other does not.

Complexity of solutions is another issue. The mechanisms used to enforce these incentives systems take up resources themselves. If the proportion of freeloaders is not high then the benefit derived from the systems may be outweighed by the resources expended to prevent them. This is analogous to hiring security guards for an amount that is greater than the value of what they have been hired to guard.

4.2.2. Trust management systems based on reputation information exchange

The other main form of inducing cooperation is trust management systems. Generally, these systems work by having nodes within the network exchange reputation information. When a node detects uncooperative behavior it disseminates this observation to other nodes which take action to avoid being affected by the node in question by changing traffic routes. In addition, some systems punish misbehaving nodes by isolating them from the network for a certain period of time in order to provide an incentive for users to cooperate. Note that although some trust management systems are also used to prevent against malicious attacks, hereafter we are only concerned with the incentives aspects.

As with the token-based incentives system, trust management systems are subject to some significant problems. The first problem is that they take up considerable resources due to the constant transmission of observation data, which serves no purpose other than to monitor node behavior. This hogs valuable CPU cycles, memory, bandwidth and battery power that could be used to send actual data.

Trust management systems also suffer from vulnerabilities due to exchanging second hand information. Nodes may falsely accuse other nodes of misbehaving or collude with each other to cheat other users on the network.

Making decisions on whom to believe in a self-organizing ad hoc network is very hard, requiring authentication as well as trust information about the accusing node. In practice this is extremely difficult to achieve, requiring either nodes which are known to be trustworthy (unpractical for ad hoc networks) or bootstrapping trust relationships which involve significant complexity and risk, and may not be possible at all for dynamic or short-lived networks [KLXH04].

These factors make it questionable whether a trust management system could be effectively implemented in reality at a reasonable cost.

In addition, there have been very few experimental tests of either type of incentives systems to date. Almost all results come from simulations, which operate under assumptions and limited conditions that do not accurately reflect reality, and most importantly do not take user behavior into account. Real life situations are invariably more complex and humans are often irrational and unpredictable, therefore, although the systems can be shown to work reasonably in simulations, real life implementations may show completely different results.

4.3. A Local Policy for Cooperation Enforcing

As we discussed in the previous section cooperation enforcing mechanisms based on distributed algorithms have a number of inherent flaws that make them difficult and undesirable to implement in practice. If badly implemented, some of them even have the potential to backfire by offering an incentive to cheat the incentives system in order to gain further benefits.
To avoid these problems we propose to tackle the problem of cooperation enforcing using a node local policy, i.e., a node takes its decisions by exploiting its own knowledge of the system. Our cooperation enforcing model is based on two main steps:

i) a node by observing the network collects information about the behavior of other nodes; This information is used to construct estimates of the other nodes reputations (i.e., a measure of their wiliness to cooperate).
ii) By exploiting the reputation indexes, a cooperation enforcing policy is then applied when the node should provide a service to the other nodes (e.g., traffic forwarding). According to the consideration in Section 7.3, our policy punishments/incentives policy is based on limiting the service rate, rather than ostracizing them from the network completely.

4.3.1. Reputation Measures

To locally construct the estimates of the other nodes' reputations we have identified two mechanisms. A first mechanism is based on the direct observation of the behaviour of neighbour nodes. This is implemented through a watchdog mechanism built-in inside the CORE mechanism. In addition, indirect estimates of the cooperation level in the different parts of the network can be obtained by exploiting the reliability indices constructed by the Reliable Forwarding mechanism presented in Section 3.3.

4.3.2. A priority-based mechanism for cooperation enforcing

To enforce nodes cooperation we propose a simple forwarding mechanism based on priority treatment. The key idea is to differentiate the quality of service toward other nodes, providing preferential or unfavourable service, according to the way they behave with others. This is done by raising/limiting the priority of traffic flows, queuing and servicing queues in different ways. Specifically, nodes can use reputation indexes to classify incoming traffic so as to slow down packets coming from less reliable nodes. In this way, packets coming from reliable neighbors are forwarded with higher priority than packets coming from neighbors with a lower reliability value.

This mechanism is based on a *priority queuing technique*. Consider the range of reputation values $[0,1]\in R$, and a partition $\{p_0,\dots,p_n\}$ of this range. Each item of the partition is associated with a priority level P_k, $1 \le k \le n$, which reflects reputation values falling in $(p_{k-1}, p_k]$ (the first partition interval covers also $p_0=0$). Given this priority scheme, incoming packets are dispatched into FIFO queues labeled each with a different priority level P_k. The dispatching policy uses the reputation value estimated for the source of incoming packets. On top of this dispatching algorithm, a scheduler chooses outgoing packets screening the queues from the highest to the lowest priority, picking up the first available packet.

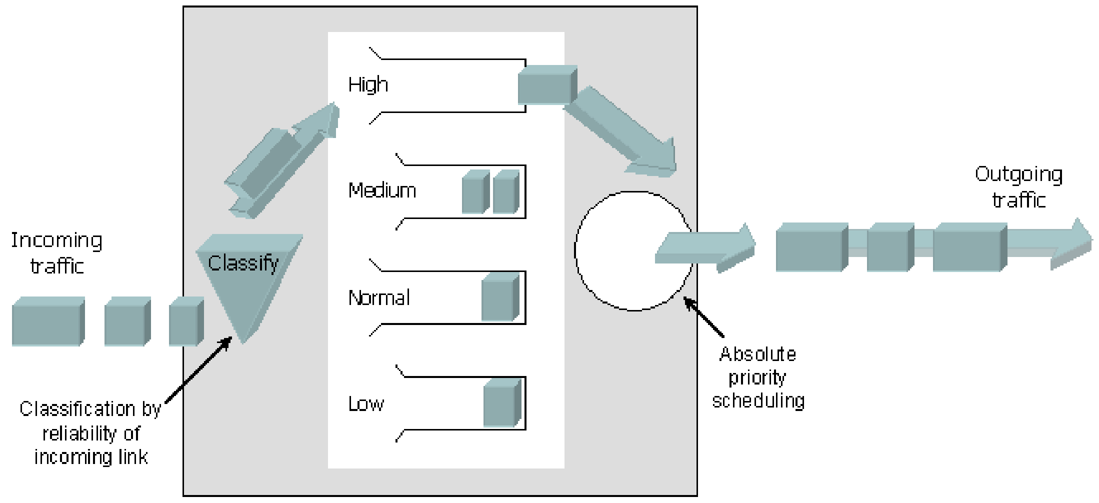

High
Medium
Normal
Low
Classify
Incoming traffic
Classification by reliability of incoming link
Outgoing traffic
Absolute priority scheduling

Figure 4.3. Priority queuing scheme for packet forwarding: packets are placed into four levels of queues: high, medium, normal, and low.

Figure 4.3 shows this mechanism in the case of four queues, with priority levels set to high, medium, normal, and low. Incoming packets are classified according to the estimated reliability of the neighbor that sent each packet, and housed in the respective queue. During transmission, the algorithm gives higher-priority queues absolute preferential treatment over low-priority queues.

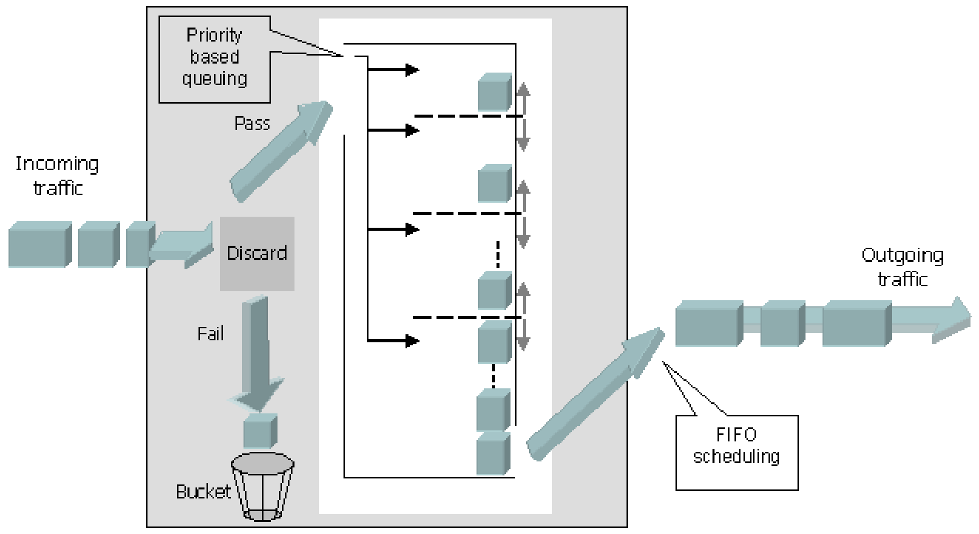


Figure 4.4. Queue management.

This scheduling mechanism comes from the fluid-based discipline called Generalized Processor Sharing (GPS), which is a well-established technique to accomplish service differentiation [PG93]. In GPS the traffic flows are divided into classes, and each class is assigned a positive weight that specifies the guaranteed minimum capacity for a class. If a particular class does not fully use its capacity, then the excess capacity becomes available for other classes passing through this node, on top of their minimum.

The less transparent the system is, the more complex it becomes for the user. A reasonable middle approach would be to have agents which handled forwarding decisions according to preset rules, based on criteria such as the battery level and the reputation values.

Because queues are of finite size, they can fill and overflow. Hence a mechanism is necessary to avoid higher-priority queues fill up, while there is still space in lower-priority ones. To this end, packets can be housed in a priority queue, which contains variable-length sub-queues (one for each priority level), and allows priority-based insertion, keeping the FIFO order (see
Figure 4.4). In this way, higher-priority sub-queues can grow up to the maximum size, eventually dropping lower-priority packets.

Forwarding incoming packets according to the priority queuing technique implies that the sender gets a quality of service proportional to its reputation level. In fact, as we previously saw, there is a strong relation between a reputation value and the behavior of the node pointed out by such a value. If the estimated reputation for a neighbour is very low, then it is quite likely that it is misbehaving. Hence, the lower the reputation of a node, the slower its traffic flows through the network.

Newly arrived nodes are assigned a medium priority. This is to give preferential treatment to nodes already belonging to the network and cooperating to its functioning. For example it could be reasonable to initiate a new node's reputation to a value about *0.7*. If the node joining the network cooperates with the others, then its reputation (and hence its priority level) is increased. This choice is motivated by the need to avoid pure selfish nodes to disappear for a while from the network (disabling all network functions) and then to join again it as new nodes (raising their priority). This initialization policy seems to penalize leaf nodes that are on the border of the network and are not required to forward traffic for the benefit of others. However, it is possible to provide a “reintegration” mechanism that periodically checks the routing table, and upgrades the reliability of nodes that are neighbors but do not appear as next hops toward other nodes. This mechanism is based on topology information that is deduced from the routing table and is subject of future work.

In case of scarce energy level, the forwarding node can decide to drop packets with lower priority, and forward only traffic classified as, for example, high and medium. In this way, a node unable to fully cooperate to the network functioning avoids being excluded from the network, because it assures service to well-behaving nodes, which in turn will continue to serve its request.

4.4. CORE: A Game Theory Analysis

As already happened for P2P systems [BAS03], game theory has proved effective in modeling cooperation problems as well as cooperation enforcing mechanisms even in the ad hoc environment. The cooperation problem is a social dilemma which has been deeply studied in disciplines like economy and social sciences since the 1950s, under a game theoretical point of view [K98]. Social dilemmas are situations in which individual rationality leads to collective irrationality. That is, individually reasonable behavior lead to a situation in which everyone is worst off than they might have been otherwise.
Since a large fraction of existing cooperation enforcement schemes are based on principles akin to decision making and economic modeling, a natural tool that emerged to be suitable for the validation of such mechanisms is game theory.

In this section we briefly review the use of Game theory for modeling cooperation issues. Then, we apply it to study the behavior of the CORE mechanism.

4.4.1. *Game theoretical modeling*

The cooperation problem in ad hoc networks is a social dilemma that can be modeled as a *game*. A game is an interaction or exchange between two (or more) actors, where each actor attempts to optimize a certain variable by choosing his actions (or "moves") toward the other actor in such a way that he could expect a maximum gain, depending on the other's response. The Prisoner's Dilemma (PD) is the game that most widely has been adopted to study an equilibrium for ad hoc systems that illustrates a conflict between individual and group rationality [K03]. The original story involves two criminals arrested under the suspicion of having committed a crime together. However, the police do not have sufficient proofs in order to have them convicted. To solve the problem the police offer a deal to the criminals. They are separately given the choice between testifying against the other (defection) or keeping silent (cooperation): the one who offers evidence against the other one will be freed. Let us quantify the game's payoffs as *a, b, c, d*. If none of them accepts the offer, they are in fact cooperating against the police, and both of them will get only a small punishment because of lack of proof (payoff *b*). They both gain. If one of them betrays the other one, by confessing to the police and the other one remains silent, the defector will gain more (payoff *a*), since he is freed, while his accomplice will go to jail (payoff *d*). If both betray, both will be punished, but less severely than if they had refused to talk (payoff *c*). The dilemma resides in the fact that each prisoner has a choice between only two options, but cannot make a good decision without knowing what the other one will do. However, independently of the other's choice, each is better off confessing than remaining silent, even if the outcome obtained when both confess is worse for each than the outcome they would have obtained had both remained silent (payoff chain $a<b<c<d$).

Such a distribution of losses and gains seems natural for the ad hoc context, since nodes are really concerned with energy consumption. Cooperating nodes not only lose resources for the benefit of others, which can be defectors, but also they are not rewarded.
By adopting a suitable level of abstraction, nodes composing the ad hoc system can be seen as interacting entities that can request or offer a service: a single node can be both a user and a provider. Considering the packet forwarding functionality, nodes have to decide whether to accept or reject any forwarding request from its neighbors [GU04]. So nodes are the game players and the game is the ``forwarding" dilemma. If they all accept (i.e. they cooperate to packet forwarding), than the overall system payoff is maximized, even if it costs some energy to nodes. On the other side, if a node accepts a forwarding request, while the others defect, then it will consume its energy without getting anything back. If all nodes reject forwarding requests, than the payoff is very low from both a system and an individual point of view, but at least nodes do not consume energy, and hence from a node standpoint it is better than being the only forwarding node. So the best strategy from an individual point of view is defection (i.e. not to forward). The following table shows the payoff matrix for two network nodes, where moves are labeled as *Acc* (accept to forward) and *Rej* (reject forwarding request), and the chain payoff is analogous to that one considered in the PD ($a<b<c<d$):

	Acc	*Rej*
Acc	b,b	d,a
Rej	a,d	c,c

As a node will be often required to forward a packet, it is possible to see the network as an infinite repetition of the forwarding game, where requests are made at every round (time is slotted and at every slot some nodes can ask to some other nodes to forward their packets). This situation is better modeled by an *iterated* (or repeated) version of the game in which players play the PD repeatedly, retaining access at each round to the results of all previous rounds, thus allowing for “punishing” defections and “rewarding” cooperation.
Each node needs a *strategy* to take a decision when a request arrives. A strategy defines a set of moves or actions a player will follow in a given game. Some deterministic strategies for the PD are [K03]:

- Tit for tat (TFT) - cooperate on the first move and then copy your opponent's last move for all subsequent moves. Thus the player begins by cooperating, and then reciprocates the opponent's move (e.g. punishes defection in the prior move by defecting). The strategy is *forgiving* in the sense of being willing to cooperate even with those who have defected against it: as soon as the opponent cooperates again, the strategy forgets about the previous defection, and cooperates.
- Tit for Two Tat (TF2T) - same as Tit for Tat, but requires two consecutive defections for a defection to be returned. The player cooperates on the first two moves, and then if the opponent defects twice in a row, the player chooses defection as the next move.
- Suspicious Tit for Tat (STFT) - Always defect on the first move. Thereafter, replicate opponent’s last move.
- Free Rider (ALLD) - Always choose to defect no matter what the opponents’ last turn was.
- Always Cooperate (ALLC) - Always choose to cooperate no matter what the opponents last turn was.

There are some situations where deterministic strategies are not convenient because the individuals operate in a 'noisy' environment. Certain actions may be misinterpreted due to random malfunctions and deterministic strategies cannot handle this feature. In the ad hoc case, packets dropping due to congestion events or lossy link can be wrongly interpreted as intentional packets discard. Instead, a stochastic strategy involves an element of randomness. One of these strategies is:

- Generous Tit For Tat (GTFT) Instead of immediately defecting after an opponent does, there is a probability that it will forgive the defection by cooperating on the next move.

A set of strategies (one for each player) leads to a Nash equilibrium if it has the property that no player can benefit by changing his/her strategy while the other players keep their strategies unchanged. If a set of nodes agrees on some specific equilibrium, it is possible to enforce cooperation by punishing deviating nodes. Strategies such as TFT and GTFT may be adopted to encourage cooperation in ad hoc networks. Most of works on cooperation enforcing uses a generalized TFT strategy that tries to balance for each node the amount of service provided with the amount of resources used.

A new trend in ad hoc research is to study user cooperation in systems where it is allowed being selfish, in order to investigate if the system reaches a Nash equilibrium without any explicit cooperation enforcing mechanism. A mathematical framework for studying user cooperation, and to define strategies leading to an optimal user behavior is provided in [SNCR03]. They propose a distributed and scalable acceptance algorithm called Generous TIT-FOR-TAT (GTFT) used by the nodes to decide whether to accept or reject a relay request. They show that GTFT results in Nash equilibrium and prove that the system converges to the rational and optimal operating point. However, this framework relies on an implicit cooperation enforcing mechanism dictated by GTFT strategy. Another solution, proposed by [FBH03], study the equilibrium when three different strategies are adopted: ALLD, ALLC, and TFT. The main distinguishing feature is that nodes make decisions based only on local information. In any case, assumptions on static routes seem too strong and do not show the applicability to real ad hoc networks.

4.4.2. Modeling Core as a *non-cooperative Game*

In the first year of the project we studied CORE behavior by using a model that takes into account both a node-centric and a network-centric perception of the interactions between nodes that participate in a MANET by using *cooperative game theory*. Although the "cooperative games" approach appears to be appropriate to model the dynamics of large coalitions of nodes forming a MANET, the main limitation of this method is that it is based on a high-level representation of the reputation mechanism that does not take into account the features of CORE. To overcome this weakness, we have developed an alternative approach based on *non-cooperative game theory* [OR97] [FT91]. In this second method we use a model that describes the strategy of a self-interested node that has to take the decision whether to cooperate or not with a randomly chosen neighbor. Under this model, the CORE mechanism can be translated into a strategy profile that can be compared to other popular strategies. Under the commonly used hypothesis of *perfect monitoring*, we demonstrated the equivalence between CORE and a wide range of history-based strategies like *tit-for-tat*. Further, by adopting a more realistic assumption that takes into account unreliable observations of nodes' behavior due to communication errors, the non-cooperative model puts in evidence the superiority (in terms of stability and robustness) of CORE over other history-based schemes.

4.4.3. Non-cooperative games approach

Hereafter, we investigated on the characteristics of CORE by modeling the interactions between the nodes of a MANET as a non-cooperative game. In the following sections we will describe how the introduction of a noise factor allows grasping the undesirable effects of using the *promiscuous mode* operation of a wireless card as a basis for the monitoring mechanism (the watchdog mechanism) and prove that the CORE strategy outperforms all other known strategies both for promoting cooperation and for the evolution of cooperation. In order to describe the interaction between nodes of a MANET and the decision making process that results in a cooperative or selfish behavior of the nodes we will use a classical game introduced by A. Tucker [P93, pages 117-118]. In the classical PD game, two players are both faced with a decision to either cooperate (C) or defect (D). The decision is made simultaneously by the two players with no knowledge of the other player's choice until the choice is made. If both cooperate, they receive some benefit (R). If both defect they receive a specific punishment (P). However, if one defects, and one cooperates, the defecting strategy receives no punishment (T) and the cooperator a punishment (S). The game is often expressed in the canonical form in terms of pay-offs:

	Player j		
Player i		Cooperate	Defect
	Cooperate	(R,R)	(S,T)
	Defect	(T,S)	(P,P)

	Player j		
Player i		Cooperate	Defect
	Cooperate	(3,3)	(-2,4)
	Defect	(4,-2)	(0,0)

Table 3. Prisoner's Dilemma payoff matrix: (a) general, (b) example.

The PD game is a much studied problem due to its far-reaching applicability in many domains. In game theory, the prisoner's dilemma can be viewed as a two-players, non-zero-sum, non-cooperative and simultaneous move game. In order to have a dilemma the following expressions must hold:

$$T > R > P > S$$
$$R > \frac{S+T}{2} \qquad (1)$$

In our model, a MANET formed by N nodes is considered as an N-player playground in which randomly, any two nodes can meet. We suppose that every node of the network has some data traffic to be sent through some source route that is the result of the execution of some routing protocol (as an example the DSR protocol). We also suppose that when any two nodes meet, at some time period t, they both need to send some data packets through each other, i.e. using each other as a relay node. Before the actual process of sending a packet, the two nodes have to take the decision whether to cooperate or defect. By cooperating a node will forward one (or more) data packet for the requesting node, whereas by defecting a node will not relay data packets on behalf of the requesting node. Instead of including an accurate description of energetic costs, topology information, possible interference and path information we will base our model on some basic economic modeling. As an illustrative and intuitive example, let us consider two players (nodes) with some letters (data messages) to send. For each letter leaving a player, a stamp (energy cost for sending one data packet) has to be used. When a letter is forwarded towards its destination the player benefit is (arbitrarily) fixed to 5: of course, the benefit for a successful communication should be higher than the (energetic) cost for sending the letter. So, for example, if two players meet and both have a letter to send, if the decision of a player is to cooperate, she will have to spend two stamps (one for her letter, and one for her opponent's letter) and eventually receive a benefit of 5 if her opponent cooperated, leading to a payoff equals to 3 in case the opponent decided to cooperate and to a payoff equals to -2 in case the opponent decided to defect. This situation can be translated in a payoff matrix which matches the one illustrated in Table 3 of the classical PD game.

Of course, it is arguable that such a simple model can represent a real MANET, but we believe that the limitations imposed by our model are greatly compensated by the consolidated theoretical results available in the literature for the prisoner's dilemma. Furthermore, we plan to extend the model in order to cope with a T-player simultaneous move game where $T < N$ thus taking into account the cooperative strategy of nodes that are part of an entire path from a data source to her selected destination.

However, the key of the model presented hereafter and any further extensions is the "willingness to communicate" assumption: during every play of the game (both in the basic PD and in the iterated PD, as we will see in the next section) both players engaged in the decision making process (cooperate or not) are supposed to have some data packets to be sent through the opponent player. As we will see later this assumption is necessary in order to implement a punishment mechanism for a non-cooperating node.

The iterated Prisoner's dilemma

One surprising feature of many one-shot games (i.e. games that are played only once) including the PD game, is that the Nash equilibrium is non-cooperative: each player would prefer to fink (defect) rather than to cooperate. However, in a more realistic scenario (e.g. a MANET) a particular one shot game can be played more than once; in fact, a realistic game could even be a correlated series of one shot games. In such iterated games an action chosen by a player early on can affect what other players choose to do later on: repeated games can incorporate a phenomena which we believe are important but not captured when restricting our attention to static, one shot games. In particular, we can strive to explain how cooperative behavior can be established as a result of rational behavior. In this section we'll discuss repeated games which are "infinitely repeated". This does not mean that the game never ends, however. We will see that this framework can be appropriate for modeling situations in which the game eventually ends (with probability one) but the players are uncertain about exactly when the last period is (and they always believe there's some chance the game will continue to the next period).
In Appendix we'll introduce in a more formal way some basic concepts related to repeated games and infinitely repeated games.

Cooperation in the Repeated Prisoner's dilemma

In the one-shot PD, the players cannot avoid choosing their dominant strategy Defect (see Table 3). In order to make the following analysis simpler, consider the following payoff matrix:

	Player j		
Player i		Cooperate	Defect
	Cooperate	(1,1)	(-1,2)
	Defect	(2,-1)	(0,0)

Table 4. Modified PD payoff matrix.

It is easy to verify that conditions (1) hold.
Even when this game is finitely repeated, because the stage game has a unique Nash equilibrium, the unique subgame-perfect equilibrium has both players defecting in every period. However, when the players are sufficiently patient it is possible to sustain cooperation (i.e. keeping "Cooperate") in every period as subgame-perfect equilibrium of the infinitely repeated game. First we will see that such cooperation is a Nash equilibrium of the repeated game. We will then show that this cooperation is a subgame-perfect equilibrium.
When an infinitely repeated game is played, each player *i* has a repeated-game strategy s_i, which is a sequence of history-dependent stage-game strategies s_i^t; i.e. $s_i = \left(s_i^0, s_i^1, \ldots\right)$, where each $s_i^t : A^t \to A_i$. The n-tuple of individual repeated-game strategies is the repeated-game strategy profile $s = \left(s_1, s_2, \ldots, s_n\right)$.

As a fundamental example, let us consider a particular strategy that a player could follow and which is sufficient to sustain cooperation. This strategy is also known as the `spiteful` strategy.

Spiteful
Cooperate in the first period.
In later periods, cooperate if both players have always cooperated.
However, if either player has ever defected, defect for the remainder of the game.

More precisely and formally, it is possible to write player *i*'s repeated-game strategy $\bar{s}_i = \left(\bar{s}_i^0, \bar{s}_i^1, \ldots\right)$ as the sequence of history-dependent stage-game strategies such that in period *t* and after history h^t,

$$\bar{s}_i^t\left(h^t\right) = \begin{cases} C, t = 0 \text{ or } h^t = \left((C,C)^t\right) \\ D, \text{otherwise} \end{cases} \tag{2}$$

First, we will show that for sufficiently "patient players" the strategy profile $\bar{s} = \left(\bar{s}_1, \bar{s}_2\right)$ is a Nash equilibrium of the repeated game. Then we will show that for the same required level of patience these strategies are also a subgame-perfect equilibrium.
Now, if both players conform to the alleged equilibrium prescription, they both play "cooperate" at *t*=0. Therefore at *t*=1, the history is h^1=(C,C); so they both play "cooperate" again. Therefore at *t*=2, the history is h^2=((C,C),(C,C)), so they both play "cooperate" again. And so on…. The path of *s* is the infinite sequence of cooperative action profiles ((C,C),(C,C),…). The repeated-game payoff to each player corresponding to this path is trivial to calculate: they each receive a payoff of 1 in each period, therefore the average discounted value of each player's payoff stream is 1.
Can player *i* gain from deviating from the repeated-game strategy $\bar{s}_i$ given that player *j* is faithfully following $\bar{s}_j$? Let *t* be the period in which player *i* first deviates. She receives a payoff of 1 in the first *t* periods 0,1,…,*t*-1. In period *t*, she plays "defect" while her conforming opponent played "cooperate", yielding player i a payoff of 2 in that period. This defection by player *i* now triggers an open-loop "defect"-always response from player *j*. Player *i*'s best response to this open-loop strategy is to "defect" in every period herself. Thus she receives zero in every period *t*+1, *t*+2,…. To calculate the average discounted value of this payoff stream to player *i* we can refer to (A.32), and substitute v_i'=1, v_i''=2, and v_i'''=0. This yields player *i*'s repeated-game payoff when she defects in period t in the most advantageous way to be $1-\delta^t(2\delta-1)$. This is weakly less than the equilibrium payoff of 1, for any choice of defection period *t*, as long as $\delta \geq \frac{1}{2}$. Thus we have defined what we meant by "sufficiently patient:" cooperation in this PD game is a Nash equilibrium of the repeated game as long as $\delta \geq \frac{1}{2}$.
To verify that $\bar{s}$ is a subgame-perfect equilibrium of the repeated prisoners' dilemma it is necessary to check that this strategy profile's restriction to each subgame is a Nash equilibrium of that subgame. Consider a subgame, beginning in period τ with some history h^τ. What is the restriction of $\bar{s}_i$ to this subgame? Denoting the restriction by $\hat{s}_i$ we have:

$$\hat{s}_i^t(\hat{h}^t) = \bar{s}_t^{t+\tau}\left(\left(h^\tau;\hat{h}^t\right)\right) = \begin{cases} C, h^\tau = \left((C,C)^\tau\right) \text{and } \hat{h}^t = \left((C,C)^t\right) \\ D, \text{otherwise} \end{cases} \tag{3}$$

We can partition the subgames of this game, each identified by a beginning period τ and a history h^τ, into two classes: A) those in which both players chose "cooperate" in all previous periods, i.e. $h^\tau = \left((C,C)^\tau\right)$, and B) those in which a defection by either player has previously occurred. For those subgames in class A), the sequence of restrictions $\hat{s}_i^t(\hat{h}^t)$ from (3) reduces to the sequence of original stage-game strategies $\bar{s}_i^t\left(h^t\right)$ from (2), i.e. for all τ and $h^\tau = \left((C,C)^\tau\right)$ we have:

$$\hat{s}_i^t(\hat{h}^t) = \begin{cases} C, h^\tau = \left((C,C)^\tau\right) \text{and } \hat{h}^t = \left((C,C)^t\right) \\ D, \text{otherwise} \end{cases} = \begin{cases} C, \hat{h}^t = \left((C,C)^t\right) \\ D, \text{otherwise} \end{cases} = \bar{s}_i^t\left(h^t\right) \tag{4}$$

Because $\bar{s}$ is a Nash equilibrium strategy profile of the repeated game, for each subgame h^τ in class A), the restriction $\hat{s}$ is a Nash equilibrium strategy profile of the subgame when $\delta \geq \frac{1}{2}$.

For any subgame h^τ in class B), $h^\tau \neq \left((C,C)^\tau\right)$. Therefore the restriction $\hat{s}_i$ of $\bar{s}_i$ specifies $\hat{s}_i^t = D$ for all $t \in \{0,1,...\}$. In other words, in any subgame reached by some player having "defected" in the past, each player chooses the open-loop strategy "defect always." Therefore the repeated-game strategy profile $\hat{s}$ played in such a subgame is an open-loop sequence of stage-game Nash equilibria. From Theorem 1 of [10] we know that this is a Nash equilibrium of the repeated game and hence of this subgame. We have shown that for every subgame the restriction of $\bar{s}$ to that subgame is a Nash equilibrium of that subgame for $\delta \geq \frac{1}{2}$. Therefore s is a subgame-perfect equilibrium of the infinitely repeated PD when $\delta \geq \frac{1}{2}$.

Complex strategies in the Iterated Prisoner's Dilemma

Axelrod and Hamilton [A84], [A87], [A81] used a computer tournament to numerically detect strategies that would favor cooperation among players engaged in the iterated PD. In a first round, 14 more or less sophisticated strategies and one totally random strategy competed against each other for the highest average scores in an iterated PD of 200 moves. Unexpectedly, a very simple strategy did outstandingly well:

TIT-FOR-TAT
Cooperate on the first period and then copy your opponent's last move for all subsequent periods

This strategy was called `Tit-for-tat` (TFT) and became the founder of an ever growing amount of successful strategies.
To study the behavior of strategies from a numerical point of view, two kinds of computation can be done.
The first one is a simple round robin tournament, in which each strategy meets all other strategies. Its final score is then the sum (not the discounted sum) of all scores done in each confrontation. At the end, the strategy's strength measurement is given by its range in the tournament.
The second type of numerical analysis is a simulated ecological evolution, in which at the beginning there is a fixed population including the same quantity of each strategy. A round robin tournament is made and then the population of bad strategies is decreased whereas good strategies obtain new elements. The simulation is repeated until the population has been stabilized, i.e. the population does not change anymore. A good strategy is then a strategy which stays alive in the population for the longest possible time, and in the biggest possible proportion. This kind of evaluation quotes the robustness of strategies.
Before the introduction of CORE as a strategy for the iterated PD, it is important to detail the computation method for ecological evolution, for example involving three strategies. Suppose that, initially, the population is composed of three strategies `A`, `B`, `C`. At generation *n* each strategy is represented by a certain number of players: $W_n(A)$ using `A`, $W_n(B)$ using `B` and $W_n(C)$ using `C`.
The payoff matrix of two-by-two meeting between `A`, `B` and `C` is computed and is thus known (see Table 3). $V(A|B)$ is the score of `A` when it meets `B`, etc…

Let us suppose that the total size of the population is fixed and constant. Let us note it Π :

$$\forall i \in [1,\infty[, \Pi = W_i(A) + W_i(B) + W_i(C) \tag{5}$$

The computation of the score (distributed points) of a player using a fixed strategy at generation *n* is then:

$$\begin{aligned} g_n(A) &= W_n(A)V(A|A) + W_n(B)V(A|B) + W_n(C)V(A|C) - V(A|A) \\ g_n(B) &= W_n(A)V(B|A) + W_n(B)V(B|B) + W_n(C)V(B|C) - V(B|B) \\ g_n(C) &= W_n(A)V(C|A) + W_n(B)V(C|B) + W_n(C)V(C|C) - V(C|C) \end{aligned} \tag{6}$$

Note that because of the subtractions the computation of *g* cannot be simplified. The total points distributed to all involved strategies are:

$$t(n) = W_n(A)g_n(A) + W_n(B)g_n(B) + W_n(C)g_n(C) \tag{7}$$

The size of each sub-population at generation *n*+1 is finally:

$$\begin{aligned} W_{n+1}(A) &= \frac{\Pi W_n(A)g_n(A)}{t(n)} \\ W_{n+1}(B) &= \frac{\Pi W_n(B)g_n(B)}{t(n)} \\ W_{n+1}(C) &= \frac{\Pi W_n(C)g_n(C)}{t(n)} \end{aligned} \tag{8}$$

All division being rounded to the nearest lower integer.

Classical results on the iterated PD, show that to bee good a strategy has to:
Not be the first to defect
Be reactive
Forgive
Be simple

The TFT strategy which satisfies all those criteria, has, since Axelrod's book, been considered to be one of the *best* strategies not only for cooperation but also for evolution of cooperation.

CORE as a complex strategy for the Iterated Prisoner's Dilemma

We are now focusing on the numerical analysis (through simulation software [MBD]) of the features presented by some specific strategies that the players of the iterated PD should follow in order to promote cooperation. Furthermore, we want to compare some of the strategies available in the game theoretic literature and known to be the "best" strategies both from a cooperation point of view and from an evolutionary point of view with the strategy derived from the CORE cooperation enforcement mechanism. Our claim is that the CORE strategy can be considered equivalent to the TFT strategy under certain circumstances (namely when the reputation buffer is of size 1). Furthermore, we will show through the evolutionary simulation that the CORE strategy outperforms over all the other analyzed strategies when the assumption of "*perfect private monitoring*" is replaced by the "*imperfect private monitoring*" assumption.

The CORE strategy[19] can be defined as follow:

CORE
Cooperate on the first move.

In each period, observe the past B opponent's moves and build a vector $\vec{b} = (b_1,...,b_k,...,b_B)$ where each element equals +1 for a cooperation and -1 for a defection.

Evaluate reputation as $reputation = \frac{1}{B}\sum_k b_k$.
If $reputation \geq 0$ Cooperate else Defect

We want to show now that the TFT strategy represents a particular case of the CORE strategy. Indeed, if we set B=1 it means that only one observation over the opponent's past moves is taken into account to build the reputation information. This implies that if the opponent cooperated in the last move her reputation will be positive and the player will chose too cooperate. Vice versa, if the last opponent's move was a defection, the reputation would be negative and the response of the player would be to defect. This is exactly what the TFT strategy implies: cooperate on the first move and do what the opponent did in the previous move.
In the following subsections we present some results obtained through evolutionary simulations using the iterated PD software available in [MBD]. The CORE strategy has been coded and added up to the list of available strategies in the software.

Simulations with the "perfect private monitoring" assumption

We present here the results the evolutionary simulation involving three strategies when the standard perfect monitoring assumption is made. Suppose that, initially, the population is composed of five strategies `tit-for-tat`, `spiteful`, `CORE`, `all-C` (cooperate always) and `all-D` (defect always). Initially, each strategy is represented by a certain number of players: 100 players using each of the mentioned strategies. As it is possible to see in Figure 4.5, after 5 generations the `all-D` strategy disappears: the winning strategies[20] are equivalent for promoting cooperation and, more important, for the evolution of cooperation. This implies that the winning strategies obtained the same payoff in a two-by-two round robin tournament and can be considered equivalent from an evolutionary point of view.

[19] The reader should be informed that hereafter we consider a limited version of the CORE mechanism in which reputation is evaluated through a simple average over the past observations made through the watchdog mechanism.
[20] Note that we are not considering the `all-C` as a winning strategy because of its history independent nature.

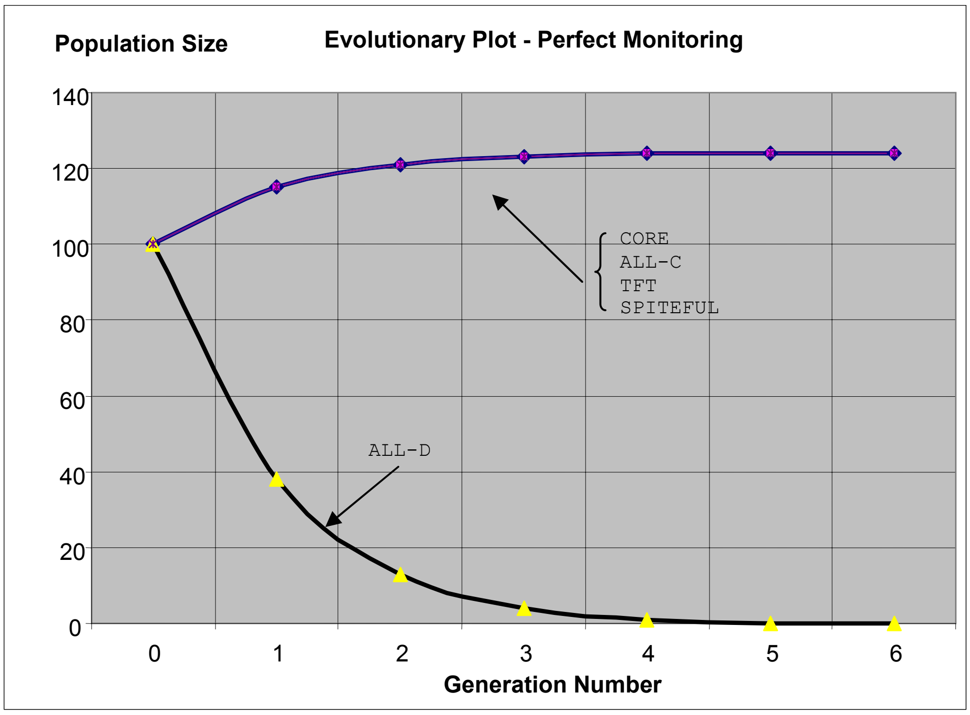


Figure 4.5. Evolutionary simulation of complex strategies for the Iterated PD with perfect monitoring.

Simulations with the "imperfect private monitoring" assumption

The majority of work in the iterated prisoner's dilemma has focused on games in a noise-free environment, i.e. there is no danger of a signal being misinterpreted by the opponent or the message being damaged in transit. This assumption of a noise-free environment is not necessarily valid if one is trying to model real-world scenarios. As a specific example, when considering interactions between two nodes in a MANET where the behavior of a node follows the game theoretical model imposed by the prisoner's dilemma, it would be interesting to consider errors due to the watchdog mechanism. The interested reader should refer to Deliverable D5 in order to understand intrinsic problems of the watchdog mechanism and the promiscuous mode operation of wireless cards. Specifically, the watchdog mechanism can be thought of as the private monitoring assumption in a two-players iterated prisoner dilemma: it is thanks to the watchdog mechanism (private monitoring) that a node (player) can infer in any period the behavior (opponent's past moves) of her neighbor and decide which actions needs to be taken (strategy).
There are different means that can be chosen to introduce noise to the simulation:

mis-implementation (when the player makes a mistake implementing its choice)
mis-perception (when one player misperceives the other player's signal or choice)

In this document we will concentrate on mis-perception noise as we believe it significantly linked to the problems introduced by the watchdog mechanism.
Kahn and Murnighan [KM93] find that in experiments dealing with prisoner's dilemma in noisy environments, cooperation is more likely when players are sure of each other's payoffs. Miller's experiments in genetic algorithms applied to the prisoner's dilemma results in the conclusion that cooperation is at its greatest when there is no noise in the system and that this cooperation decreases as the noise increases [M96]. Some ideas to promote cooperation in noisy environments

have been posited by Axelrod; these include genetic kinship, clustering of like strategies, recognition, maintaining closeness when recognition capabilities are limited or absent (e.g. limpets in nature), increasing the chance of future interactions (certain social organizations, hierarchies in companies etc.), changing the pay-offs, creating social norms where one learns cooperation. Hoffman [H00] reports that results are sensitive to the extent to which players make mistakes either in the execution of their own strategy (mis-implementation noise) or in the perception of opponent choices (mis-perception noise).

In particular, cooperation is vulnerable to noise as it is supported by conditional strategies. For example, in a game between two TFTs, a single error would trigger a series of alternating defection. A number of authors confirm the negative effect of noise of TFT and find that more forgiveness promotes cooperation in noisy environments [M88], [BKS91].

We executed an evolutionary simulation involving 5 strategies when misperception noise was taken into account: we decided to set the noise to the value of 10% and we took the average population size over 5 simulation runs. 100 players for each of the following strategies competed in a round-robin tournament: `tit-for-tat`, `spiteful`, `CORE`, `gradual` and `soft-majo`[21]. As it is possible to observe in Figure 4.6, the `CORE` strategy outperforms and results to be the most evolutionary stable and robust strategy among all the population (we believe though that exceptions especially constructed in which performances of `CORE` are not so outstanding are possible but are seldom and not easy to obtain).

The reason why CORE performs better than the other strategies when the imperfect monitoring assumption is made can be explained as follows: by adopting the CORE strategy, a node base her decision of whether to cooperate or not using a certain amount of observations made on the opponents past moves as defined by the B parameter. Thus, the reputation measure evaluated by the node takes into account more than one observation and is less sensible to any mis-perception noise. Furthermore, in its advanced version (which has not been implemented in the simulations, though), the CORE strategy also weights the past B observations giving more relevance to past observations than recent ones. It is intuitive to realize that a transient misbehavior is filtered out by the reputation mechanism that makes CORE more flexible and "forgiving" in presence of temporary misbehavior or a momentary high percentage of noise. A specific example of such a situation can be found when thinking of a communication between nodes of a MANET in presence of obstacles or high interference.

In Figure 4.7, two populations of 100 members adopt respectively the CORE and the TFT strategy. The noise value has been set to 20% and 10 rounds of simulations have been executed in order to take average values of the evolution of population sizes. It is possible to observe that both strategies are evolutionary stable, in the long run; however, CORE is the winning strategy as the population size of players adopting it increases at each new generation. Furthermore, Figure 4.7 shows that the reputation buffer size (B) and both the stability condition and population size are directly related. As B increases, stability is reached at a lower generation number (i.e. earlier) and the population size of players adopting the CORE strategy grows faster. We believe however that these interesting results have to be evaluated in an analytical way: the fine-tuning of CORE parameters (such as B and the frequency at which observations are made) would require a laborious empirical study if carried out only by means of evolutionary simulations. We plan to analyze the CORE strategy in our future work taking as a starting point the analysis of the SPITEFUL strategy.

[21] The `gradual` and `soft-majo` strategies are described in [23].

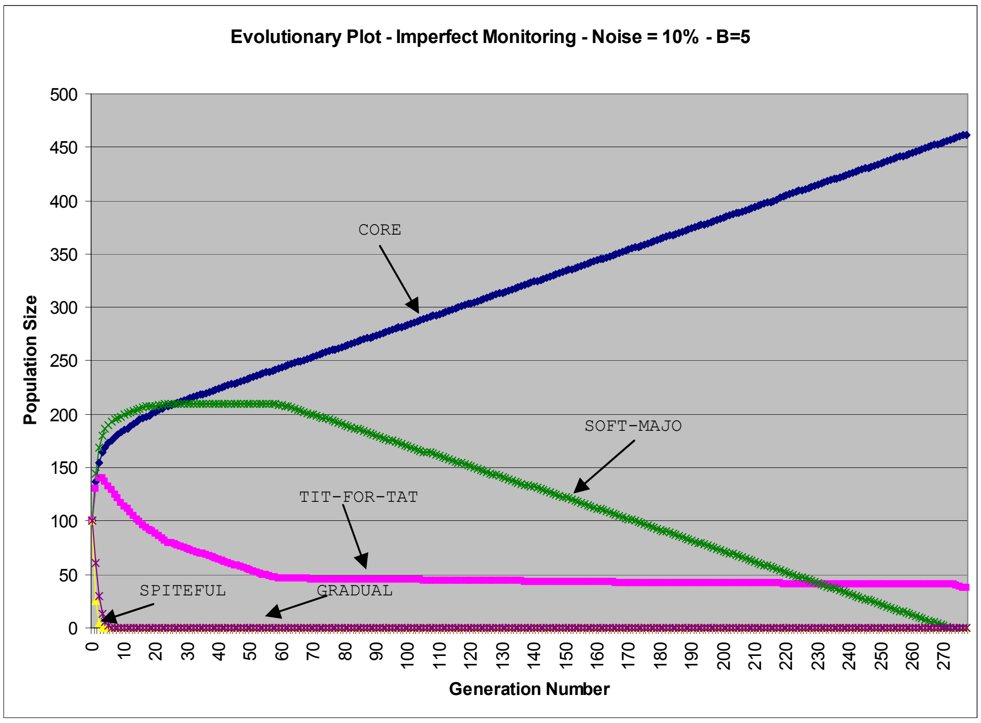


Figure 4.6. Evolutionary simulation of complex strategies for the Iterated PD with noise.

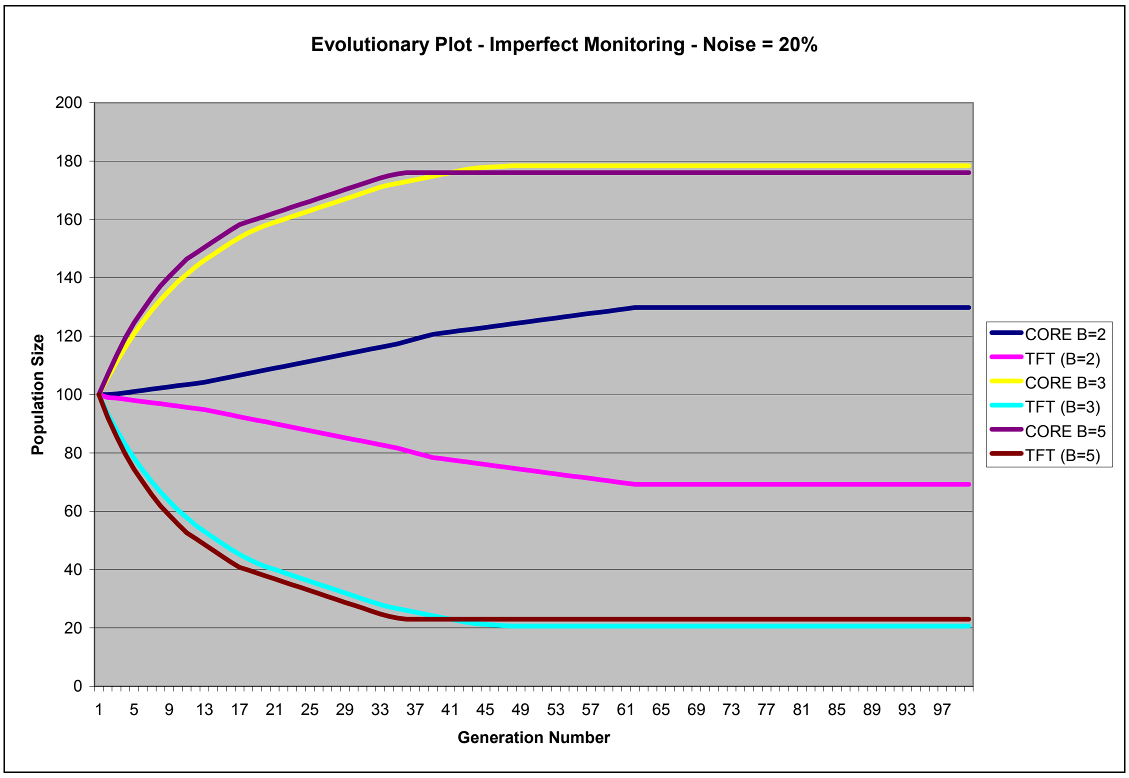


Figure 4.7. Evolutionary simulation: CORE vs. TFT with different Buffer sizes (*B*).

4.4.4. Discussion

In this section we presented a “non-cooperative” approach to assess the features of the CORE cooperation enforcement mechanism. The results provided by the “non-cooperative” approach better characterize CORE with respect to other mechanisms in a realistic setting. We were able to demonstrate the equivalence between the “TFT” strategy and CORE: precisely TFT can be thought of as a special case of the CORE strategy. Moreover, in order to represent a more realistic scenario for the execution of a (infinitely) repeated game we introduced a noise factor, which affects the players (nodes) perception of the opponents past moves. The “imperfect private monitoring” assumption allowed to model in a realistic way the watchdog mechanism used by CORE (and by most of the available cooperation enforcement mechanisms) as it is known to be particularly unreliable. We showed through evolutionary simulations that the CORE strategy outperforms all other studied strategies in a noisy environment for its stability and robustness. As a future research, we plan to extend the system model in order to take into account multiple players involved in the same decision making process in order to overcome the limitations due to a pair-wise interaction.

4.5. Core : reputation-based cooperation enforcement mechanism for mobile ad hoc networks

4.5.1. Objectives

The objectives of the project are to define, implement and evaluate a mechanism of cooperation enforcement suitable in a MANET environment. In section 4.5.2 we give an overview of such mechanism describing how it has been designed, what are the problems that we faced during the design and how those problems have been addressed. Section 4.5.4 explains how we set up a real-life test bed that will allow us to evaluate the selected mechanism of cooperation enforcement as well as other critical aspects of the operations of a MANET such as the performances of different routing algorithms implementations.

4.5.2. CORE Design and Implementation

CORE recalls

The cooperation enforcement mechanism proposed for the MobileMan architecture is the CORE mechanism. CORE [MMC02] is a collaborative monitoring mechanism based on reputation that strongly binds network utilization to the correct participation to basic networking function like routing and packet forwarding. CORE is being implemented as a Linux user-space daemon that runs on all the nodes of a MANET and can possibly be used with different routing protocols. In the future CORE will be able to store reputation information in a local storage accessible from different layers of the MobileMan stack in order to help inter-layer optimization.
First version of CORE implementation only tackles packet forwarding function.

Interactions between CORE and the Routing Function

The needs for an interaction between CORE and the routing layer should appear clear with following example.
Let’s examine the case described in figure 4.8: node A monitors the behavior of its neighbors using CORE, thus it has its WLAN card set in promiscuous mode and can inspect all the packets within its transmission range.
A wants to send a packet P to node C, which is outside node A transmission range. B is one of A neighbors and acts as a relay for node A traffic toward node C. Let’s assume that A is trying to communicate with C for the first time.

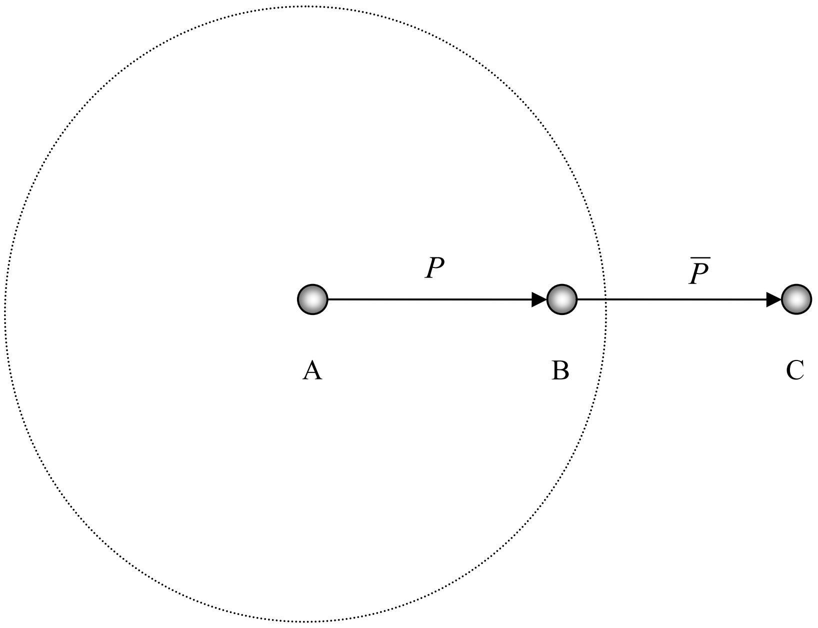


Figure 4.8: Node B is relaying a packet

In such case, with respect to the following notation:

$IP_s(P)$ = Source IP address of packet P.
$IP_d(P)$ = Destination IP address of packet P
IP_A = IP address of node A
$MAC_s(P)$ = Source MAC address of packet P
$MAC_d(P)$ = Destination MAC address of packet P

Packet P will have the following attributes:

$$IP_s(P) = IP_A$$
$$IP_d(P) = IP_C$$
$$MAC_s(P) = MAC_A$$
$$MAC_d(P) = MAC_B$$

If the MANET is operated by a reactive routing algorithm such as DSR, when A sends packet P, A knows that it will be routed to C by node B as the complete route information is included in packet P. At this point A expects to see B sending a packet $\overline{P}$ with:

$$IP_s(\overline{P}) = IP_A$$
$$IP_d(\overline{P}) = IP_C$$
$$MAC_s(\overline{P}) = MAC_B$$
$$MAC_d(\overline{P}) = MAC_C$$

Node A adds an entry containing this information to an expectation table kept in the device memory and sets a timeout for this entry. TCP sequence number of the packet is part of the entry of the expectation table to allow complete identification of the packet.

If node A observes packet $\overline{P}$ before the timeout expiration, it deduces that node B is correctly participating to the packet forwarding function and a value of +1 is passed to the function that calculates the reputation of node B, otherwise A deduces that B is not participating to the packet forwarding function and a value of -1 is passed to the function.

The timeout duration choice is critical as a too little value would result in an incorrect understanding of neighbor behavior. On the other hand a too long timeout value would cause the expectation table to grow in an exaggerated way with consequent memory shortage risks.

The reputation function includes a buffer of the last *B* observations and the reputation of a node is calculated as the average of the values contained in the buffer.

This way, node A maintains a reputation table containing the reputation values of all its neighbors.

If the MANET operates a pro-active routing algorithm such as OLSR, some differences must be taken into account.

When A sends packet P, it has no a-priori knowledge of the route the packet will follow. Node A will expect to see a packet $\overline{P}$ having:

$$IP_s(\overline{P}) = IP_A$$
$$IP_d(\overline{P}) = IP_C$$
$$MAC_s(\overline{P}) = MAC_B$$

A will have to rely on B to forward the packet to the right next hop. Secure routing is needed to avoid traffic subversion.

In both cases, node A needs to be aware of its neighborhood in order to keep up-to-date reputation information for each one of its neighbors. This can be achieved by getting neighborhood information from the routing protocol, which would make CORE dependent on the implementation of the routing protocol.

Dependency on the implementation involves a loss of generality that we would like to avoid. The same result can be obtained by integrating CORE with a neighbor discovery mechanism.

Neighbor discovery is generally achieved by the use of broadcast “hello” packets by all the nodes of a network. This involves a considerable network overhead. Moreover, all the nodes of the network need to process such “hello” packets and this can be critical in the case of Ad-Hoc networks, were energy saving is a main requirement.

After those considerations it should be clear that the optimal solution consists of a cooperation mechanism that is not dependent on the routing protocol and that is not based on a neighbor discovery realized through broadcast “hello” packets.

In the following, we describe an approach that is designed to be independent on the routing protocol and that performs neighbor discovery only when it is needed.

Following this approach, a node still listens to all the traffic within its transmission range by setting the WLAN card in promiscuous mode and according to the content of the packets that it inspects, constructs neighborhood information and updates neighbors' reputations.
A node can observe three different types of packets that trigger different actions, those use-cases are explained in the following.

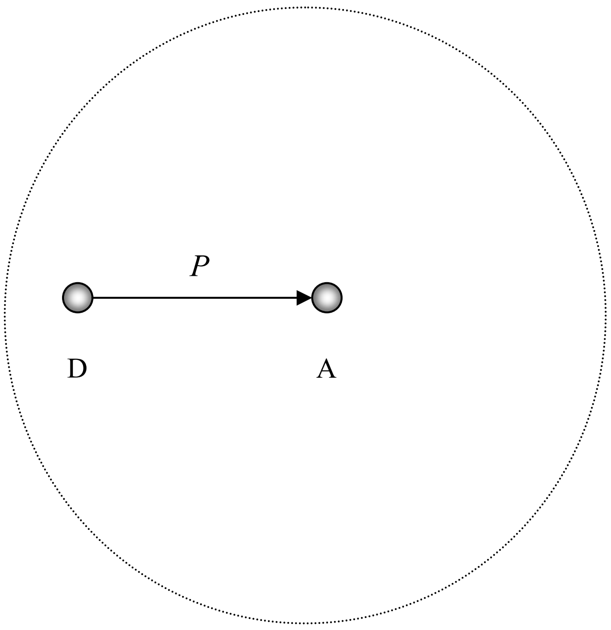


Figure 4.9: A receives a packet

In case described by figure 4.9, A observes packet P which has the following attributes:

$IP_s(P) = irrelevant$

$IP_d(P) = irrelevant$

$MAC_s(P) = MAC_D$

$MAC_d(P) = MAC_A$

IP_s is irrelevant because the fact that D is or not the logical source of packet P has no influence on node A actions. In the same way, the fact that A is or not the logical destination of the packet has no consequences.
In such case, node A can record MAC_D as the physical address of one of its neighbors.

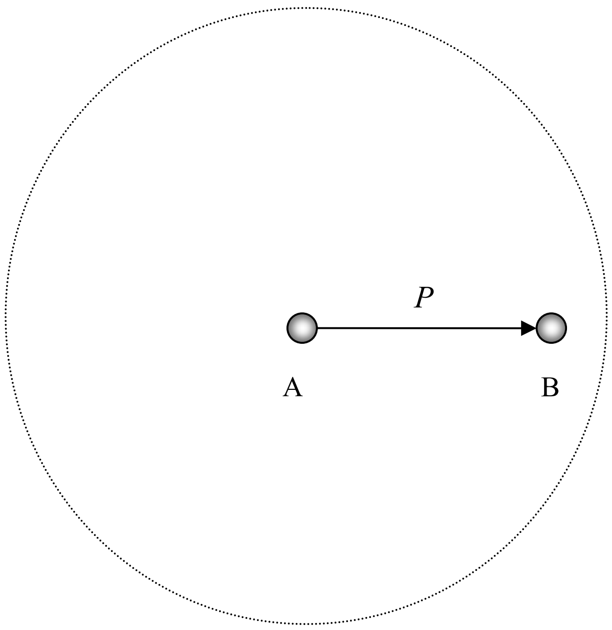


Figure 4.10: A sends a packet

In case described by figure 4.10, packet P has the following attributes:

$$IP_s(P) = irrelevant$$
$$IP_d(P) = IP_X$$
$$MAC_s(P) = MAC_A$$
$$MAC_d(P) = MAC_B$$

In this case, node A is the physical source of packet P. In order to take further decisions on what to do with this packet, node A needs to know if the logical destination of P is one of its neighbors.
A generates then an ARP request for the logical address IP_X and adds packet $\overline{P}$ to its expectation table with:

$$IP_s(\overline{P}) = IP_s(P)$$
$$IP_d(\overline{P}) = IP_X$$
$$MAC_s(\overline{P}) = MAC_B$$

For such packet, node A will set a timeout as explained above.
On timeout expiration, node A first checks if the packet associated to the timeout is still in the expectation table. If this is the case, node A checks if a ARP request for IP address IP_X has completed. If this is the case and the couple of addresses returned by the ARP reply is (MAC_B, IP_X), this means that the logical destination of P is a neighboring node, A records MAC_B as the physical address of a neighbor. If the logical destination is not a neighbor of A, A

concludes that packet P needed a relay that did not take place; a value of -1 is passed to the function that calculates the reputation of node B.

Data collected through ARP is cached to avoid useless duplicated physical address resolutions. Unrequested ARP replies are discarded for security reasons.

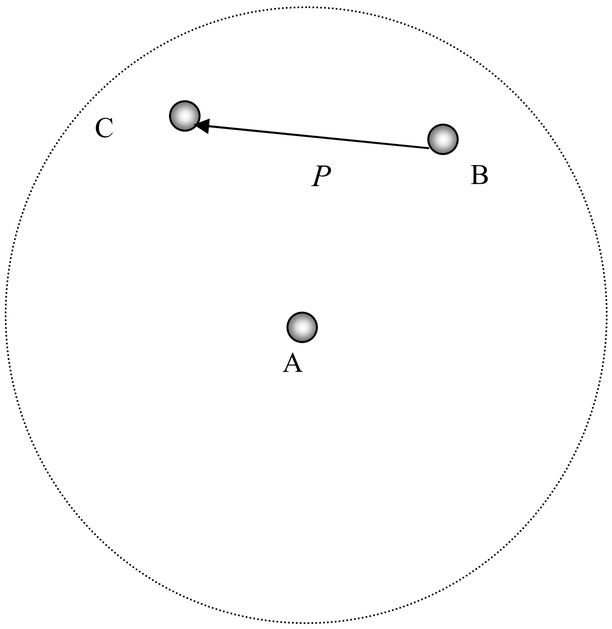


Figure 4.11: A sees two nodes talking

In case described by figure 4.11, packet P has the following attributes:

$$IP_s(P) = irrelevant$$
$$IP_d(P) = irrelevant$$
$$MAC_s(P) = MAC_B$$
$$MAC_d(P) = MAC_C$$

When node A observes such packet it verifies if its expectation table contains a packet having $IP_s(P), IP_d(P), MAC_B$ plus the TCP sequence number of packet P. If this is the case, A can deduce that B is correctly effectuating a relay. Packet P is deleted from the expectation table and the associated timeout is reset. Finally a value of +1 is passed to the function that calculates the reputation of node B.
If packet P is not in the expectation table, node A records MAC_B as the physical address of one of its neighbors.

A side effect of this approach to the watchdog problem is that the neighborhood information is not exhaustive. A silent neighbor will go undetected but this does not harm the CORE functions. The situation where a node moves out of transmission range will go undetected as well. This is

not necessarily a drawback because if such node re-enters the transmission range, the previously collected reputation information will still be considered valid.
Using ARP to resolve IP addresses still generates some overhead in terms of network utilization and processing resources, but as ARP requests are sent only when it is needed, these overheads results lower than in the case of neighbor discovery mechanisms based on "hello" packets broadcast.
This approach still relies on the secure routing hypothesis, assuming correct operation of the routing function by all the nodes of the MANET.

4.5.3. CORE Software Architecture

An important step of the validation of CORE cooperation enforcement mechanism is the development of a software prototype that realized the CORE mechanism functionalities. In the following we will refer to this software implementation as the "CORE watchdog". The architecture of the CORE watchdog is shown in figure 4.12.

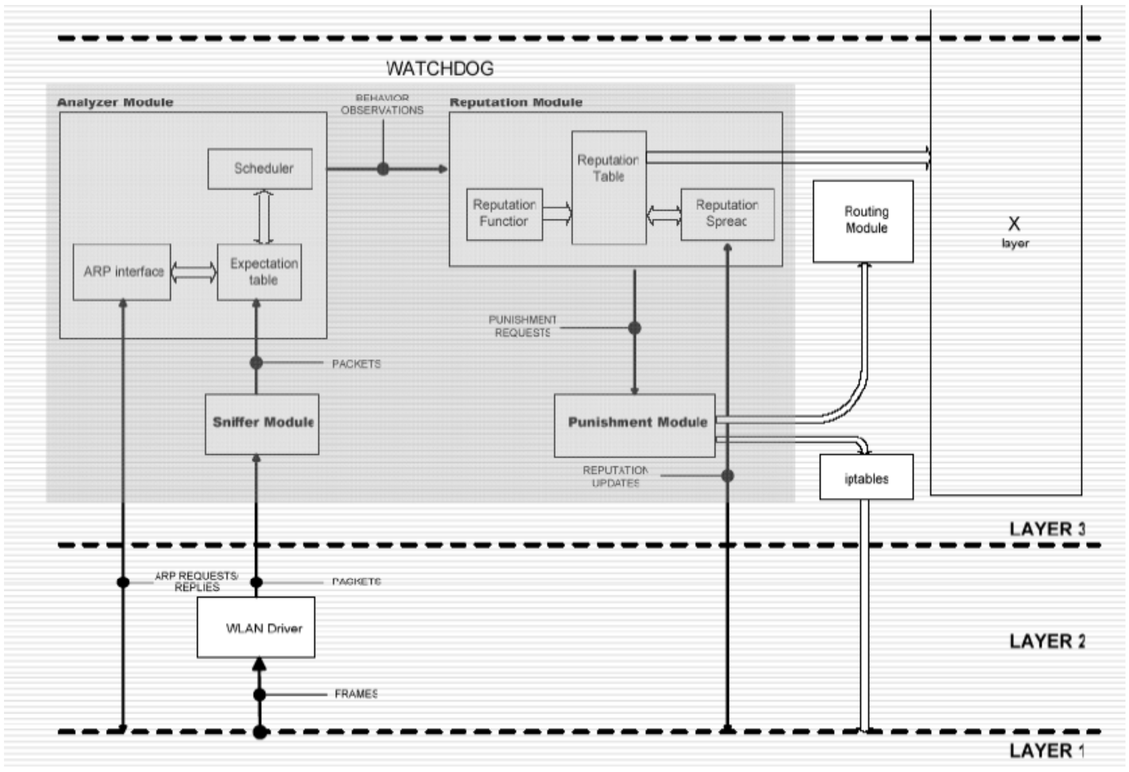


Figure 4.12: CORE watchdog architecture

The implementation is made of the following modules:
Sniffer Module: monitors the packets that pass across layer 2 of the TCP/IP stack. As the WLAN adapter is set in promiscuous mode, the sniffer module has access to all the packets and not only to the packets that are directed to the node itself. The relevant fields of packet headers are passed to the analyzer module for further analysis in the form of packet descriptors.
Analyzer Module: Receives packet descriptors from the sniffer module. Analyzes those descriptors to deduce whether the neighbors are being cooperative or not.

The analyzer module includes an expectation table where it stores some fields of the packet descriptors that correspond to packets for which forwarding is expected by a neighbor. Those relevant fields are stored in the expectation table until a timeout, handled by a scheduler API, is triggered. If the watchdog observes that the packet forwarding happened before the timeout expiration, it deduces that the node that forwarded the packet has been cooperative. If the packet is not forwarded before the timeout expiration, the node that was expected to forward the packet is treated as selfish. The analyzer module features an ARP interface that is needed to perform some basic neighbor discovery functions.
Reputation Module: Uses a reputation function to calculate values for neighbors according to behavior observation coming from the analyzer and stores those values in a reputation table. When the reputation of a neighbor falls below a given threshold, it issues punishment requests to the punishment module.
Punishment Module: Punishes selfish neighbors by blocking packet forwarding through iptables framework.
CORE watchdog implementation is open to cross-layer extensions.

4.5.4. Test Bed

CORE and Promiscuous Mode

Each MANET node implementing CORE performs RF monitoring, thus it captures all the packets that are sent within its wireless channel. To do this the WLAN card needs to be put in the so-called promiscuous mode. In this mode the card passes all the packets received to upper layers for further processing. The ability of putting the card in promiscuous mode should be supported by the firmware on the cards as well as by device drivers.
Currently many 802.11 WLAN cards do nothing when put in promiscuous mode.
As one of the goals of the project is to evaluate CORE and MANET performances in a real-life test bed, we evaluated existing WLAN cards looking for one that works correctly in promiscuous mode. We ended up choosing the Dell TrueMobile 1150 WLAN card. The Dell TrueMobile 1150 needs the Orinoco driver to be operated and works in promiscuous mode out of the box. Please note that a WLAN card that can be operated in promiscuous mode is of great interest even outside the scope of the CORE implementation, as it can be used together with WLAN sniffers such as Ethereal and Airopeek to monitor the general behavior of a MANET.
Once the card is set in promiscuous mode, the packets are captured using the pcap C libraries. Those libraries are available for Windows OS as well, making the porting effort of the CORE mechanism to Windows OS acceptable. Pcap libraries have some known limitations when used to sniff WLAN traffic, but they fit well in the case of pure Ad-Hoc networks.

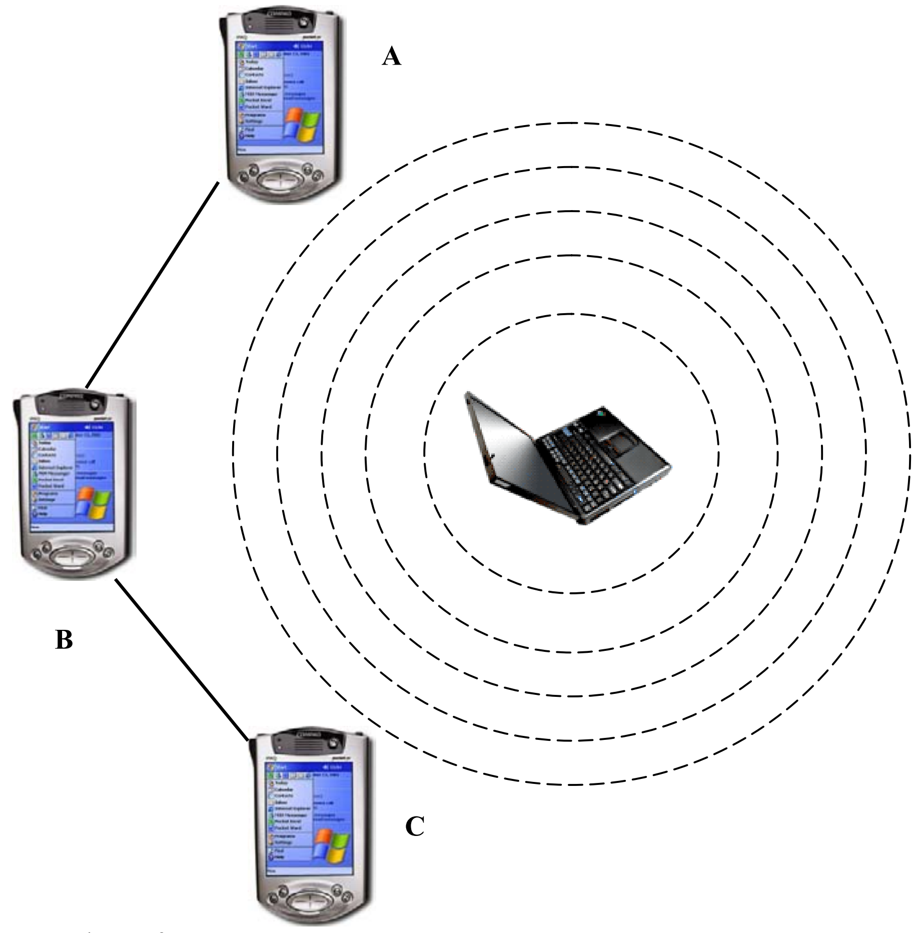


Figure 4.8: Testing Infrastructure

Test Bed hardware and architecture

In order to perform real-life testing of CORE mechanism, we selected appropriate hardware and routing algorithm implementation and deployed a simple but scalable testing environment. We decided to run our experiments using OLSR as a routing protocol. The nodes of the test network are iPaqs as those devices incarnate the typical node of a MANET in terms of reduced battery life and high mobility. The WLAN cards are the ones mentioned in paragraph 4.3.3.1.

In Figure 4.8, we show a typical configuration that is used to test selfish nodes detection: Node B is expected to act as a relay. If node B does not forward packets in the case of a communication from node C to node A, Node C should detect such selfishness and punish node B. The laptop in the figure is equipped with a WLAN card set in promiscuous mode and runs sniffing software to inspect the global testing network behavior.
During the tests selfish behavior of node B is simulated by blocking the packet forwarding through the usage of the iptables Linux command.

4.6. Cooperation: a Social Science Perspective

Under what conditions does an individual voluntarily cooperate to pursue a common goal, such as sharing his MobileMAN device with other users, as to allow the system to function? Indeed, the viability of ad hoc networks in general and the MobileMAN paradigm in particular depends, among other things, on its users' willingness to cooperate.
However, as theory and extensive empirical research shows, cooperation, i.e. people's voluntary participation to the provision and maintenance of a collective good, cannot be taken for granted for there are often clear discrepancies between individual goals and collective interests. As this review of collective action theory shows, people's willingness to cooperate depends on a number of factors that need to be taken into account in the development of any ICT whose viability is contingent upon cooperation.
As will become clear in the following pages, cooperation as a social phenomenon stands at the centre of lively academic debates, whereby a clear distinction ought to be made between 'cooperation optimists' and 'cooperation pessimists'. This paper starts with a summary of the key arguments brought forward by 'cooperation optimists' to be followed by a brief discussion of the most popular paradigms of 'cooperation pessimists'. From there, the discussion will move to some factors that are generally considered to affect people's willingness to cooperate. The paper will conclude with a discussion of the relevance of collective action theory for MobileMAN.

4.6.1. Key Theories and Concepts

Cooperation theorists may be divided in so-called 'collective action optimists' and 'collective action pessimists'. By 'collective action optimists' we refer to those social scientists, who assume that whenever cooperation is required for the mutual benefit of a group of people, it will naturally occur. Participation optimism originates from orthodox group theories prevailing in political science in the 1950s. In those years political theorists believed that the existence of a collective interest constitutes a sufficient motive for joint action, and that, if given a chance, people would try to influence decisions that affect their lives. Failures to live up to these expectations were considered abnormalities, signs of individual or systemic pathologies22.
However, low participation in elections, voluntary organisations, and collective action in general, led political scientists to question the validity of these assumptions already during the 1960s. The costs of participation were recognised as a factor that may induce individuals to take a 'free ride' on other people's efforts instead of sharing the costs or burden of cooperation. This has lead in the late 1960s to an increased pessimism in social science about people's inclination towards voluntary cooperation. Three distinct paradigms have been particularly influential in supporting theories about people's limited capacity to further common interests: the 'logic of collective action', 'prisoner's dilemma' and 'the tragedy of the commons'. These three paradigms share some fundamental views about the inherent conflict between individual and group interests, and had powerful influence in academic and policy circles.

The logic of collective action

'The logic of collective action' is the title of a book written by Olson in 1965, which even today may be considered the touchstone of collective action theory. Olson's theory is often used to demonstrate that rational individuals are unlikely to participate in a group endeavor to pursue a

[22] For a critical summary of orthodox group theory see Nagel (1987) and Olson (1965)

common goal. By 'collective action', Olson refers to group efforts to further common interests. His logic therefore encompasses almost all acts of cooperation aimed at goals shared by a group of people. These goals may relate to a tangible good, or to immaterial benefits, but they all have in common that if the goal is achieved, everybody benefits from it, regardless of whether he or she contributed to its provision. Economists, including Olson, refer to these sorts of group goals –characterized by jointness of supply and impossibility of exclusion– as 'public goods'. The problems related to the non-excludable nature of public goods and that economically rational individuals would not voluntarily contribute to pay for them was well understood in economics already before Olson. This author, however, recognized the link between collective action and public goods and that all group goals and group interests are subject to the same dilemma. Olson realized the relevance of the theory of public goods for the analysis of collective action, thus exporting its inherent logic to other social sciences. Contrary to Hardin's 'Tragedy of the Commons' and to the 'Prisoners Dilemma', Olson does not deny the possibility of rational individuals' pursuing a public interest; rather, he offers a radically different account of the logical basis of cooperation. As will be discussed later, size and other group characteristics are considered by Olson of central importance in determining an individual's attitude towards cooperation.

Another factor that often explains cooperation is coercion. For Olson, however, coercion is only one instance of a broader group of phenomena he calls 'selective incentive', which are material or social rewards specifically oriented towards those who contribute to a collective action. Collective action, according to Olson, is always accompanied by private incentives to reward contributors or to punish non-contributors. Selective incentives are not oriented towards the group as a whole like a collective good but operate selectively, towards the individuals in the group. They must be selective, so that those who do not cooperate can be treated differently from those who do. Only selective incentives or private benefits would stimulate a rational individual to act in a group-oriented way. Selective incentives are one of Olson's central themes and may be considered his 'simple and sovereign' theory of collective action (Marwell and Oliver 1993:5).

Olson's logical theory should be known to those who are excessively optimistic about people's willingness to cooperate. However, it should not be embraced indiscriminately either. As argued by Marwell and Oliver:

Free riding is a real problem. And yet collective goods are everywhere provided. […] Any reasonable theory must account for these phenomena, as well as for the equally obvious fact that many collective goods ardently desired by some groups, or even a whole population, never come to pass. (1993:6)

There are many factors that may explain collective action other than those discussed by Olson. First of all, as pointed out by Melucci (1995:18), it is necessary to overcome 'the Olsonian individualism'; 'the naïve assumption that collective phenomena are simply empirical aggregations of people acting together' needs to be discarded. Secondly, by also considering non-material rewards as acceptable selective incentives, and by recognising that also 'extra-rational motivations' (such as moral motivations and self-realisation) may determine individuals' participation to collective action, it is possible to recognise many more situations under which it may occur23.

[23] The importance of extra-rational motivations is recognized by Hardin, who dedicates a whole chapter of his book on collective action to this subject (Hardin 1982:101-124). This issue is also extensively discussed by Kollock (1999) and Rheingold (1993) with specific reference to cooperation in the cyberspace.

Prisoners' Dilemma

The Prisoners' Dilemma originates from mathematical game theory, which was one of the dominant frameworks for analysing social interactions in the fifties and sixties. It was discovered by Flood and Dresher, who were concerned with testing solutions for non-cooperative games. It was named "Prisoners' Dilemma" only later, by A.W. Tucker, a game theorist of Princeton University (Hardin 1982). The Prisoner's Dilemma shares with Olson's theory of collective action its generality and its apparent power in providing a solid basis for a profoundly disturbing conclusion –that rational people cannot achieve rational collective outcomes. The Prisoners' Dilemma suggests in a clear manner that it is impossible for rational people to cooperate, a conclusion that bears directly on fundamental issues in ethics and political philosophy. Indeed,

The paradox that individually rational strategies lead to collectively irrational outcomes seems to challenge a fundamental faith that rational human beings can achieve rational results. (Ostrom 1990:5)

Like Olson's logic of collective action, the Prisoners' Dilemma has been applied to a broad spectrum of situations. Several theorists have relied on this argument to provide the essentials of a theory of state, which would be needed above all to enforce contracts and punish deviants, so that social order can be maintained. It was also frequently used to explain the depletion of common pool resources and the failure of groups to provide or maintain public goods.

Its application to real life situations has been strongly criticised by many scholars24. In fact, not only does empirical evidence prove differently, but the way the game is structured has generally little in common with reality. Runge (1992), for example, calls attention to the fact that the game represents a special case of joint action that can only be understood if one recognises the structure of the game as a function of the institutional environment in which it is embedded. Those who see in the Prisoner's Dilemma an inevitable human tendency tend to confuse cause and effect, and that the result of the game is just an artefact of the way in which it is set up. If the rules for exacting confessions from apprehended prisoners are different, they will also have different strategies and better reasons to cooperate. Thus, as argued by Bromley:

It is essential to understand that the institutional structure of any game (or life situation) reflects the prior social purpose to be served by the human interaction under consideration … The existing institutional structure reflects, among other things, prevailing cultural and social norms regarding individualism and its relation to collective notions. In that sense we can say that people behave (or choose) in an institutional context –not a very surprising observation really. (1992:6)

The applicability of the Prisoners' Dilemma to real life collective action problems is also questioned by Wade (1988) for whom two key assumptions must hold if a situation is to be plausibly modelled as a Prisoners' Dilemma and its pessimistic conclusions drawn. The first is that the players choose in ignorance of each other's choices, that they have no contact with each other, cannot negotiate themselves and change the rules and have no previously established shared values or moral codes of behaviour. The second is that each player chooses only once before the payoffs are received, and so cannot change his mind upon finding out what the other has done. Quite obviously, where the situation is an enduring and recurrent one, the logic changes and individuals may find it convenient to cooperate. In particular, if they know what others have chosen and can alter their choice accordingly, the rational strategy may be one of 'conditional cooperation', or 'cooperate first, defect if the others defect' (Wade 1992:203).

[24] See Bromley (1992), McCay and Acheson (1987) and Ostrom (1990) for comprehensive and empirically grounded critiques to the application of the Prisoners' Dilemma.

The Tragedy of the Commons

The still widely used metaphor 'tragedy of the commons' owes its origins to an article by Garrett Hardin appeared in Science in 1968. Hardin (1968) was not exclusively concerned with common property resources, but with what he names 'no technical solution problems' in general. These include a broad array of problems such as population explosion, air pollution, deforestation, industrial waste control, and so on. To make his point about the inevitable conflict between individual behaviour and collective interest, Hardin invites the reader to picture a pasture open to all. In such a situation a rational herdsman will seek to maximise his gain by adding more and more animals to his herd. The tragedy is caused by the fact that each herdsman will act the same way as 'each being is locked into a system that compels him to increase his herd without limit in a world that is limited' (Hardin 1968: 1244).

The powerful impact of this article may partly be explained by the time and socio-cultural context in which it was published. It was in those years that the western world became suddenly aware of the dramatic consequences of an unconcerned use of natural resources that was rapidly leading to their depletion and to an irreversible loss of biodiversity.

4.6.2. The conditions for cooperation

The role of group attributes

The last pages presented the essence of two convergent perspectives about the possibility of collective action. The first is characterised by an overly simplistic optimism about people's capacity to cooperate, and the second by sweeping pessimism. Over the last decades these extreme positions have been gradually abandoned and there has been a significant advance in understanding the conditions under which collective action emerges.

The point of departure for most research on determinants for collective action still remains Olson (1965), who explains collective action by focusing on group attributes. In this section we shall discuss the importance of group attributes by focusing on five key issues, namely group size, heterogeneity, interdependence, the role of organisations, and the importance of 'community'.

Group size

One of the most controversial issues in contemporary literature on collective action is related to the effect of group size on the likelihood of group action. The focus of many scholars on this variable is partly due to the influence of Olson (1965), whose main conclusion about collective action is stated in terms of group size.

According to Olson (1965) a clear distinction ought to be made between the behaviour of individuals depending on some basic characteristics of the group to which they belong. To explain this point he uses two parallel typologies. The first one is based on group size whereby a distinction is made between small, intermediate, and large groups. The second typology differentiates between 'privileged' groups and 'latent groups'. A privileged group is defined as a group in which each of its members or at least some of them have an incentive to see that the collective good is provided, even if he has to bear the full burden of providing it himself. In such a group the collective good may be obtained even without any group organisation or coordination (Olson 1965:50). The concept of 'privileged groups' relates to the assumption of groups being heterogeneous and also draws attention to the role of organisations, two important issues that will be discussed further below. A latent group is defined as a very large group distinguished by the fact that whether a specific member contributes or not to provide the collective good, will not significantly affect other members. Therefore none has any reason to act. The distinction between latent and large groups is less clear and the two terms are almost used interchangeably. Although Olson is primarily interested in large or latent groups, he also discusses the behaviour of small groups and intermediate groups, which are more relevant for the social validation of MobileMAN.

According to Olson, small groups may be able to provide themselves with a collective good simply because of the attraction the good may have on the individual group member. The greater the interest of any single group member, the greater the likelihood that the member will get such a significant portion of the total benefit from the collective good that he will gain from seeing that the good is provided, even if he has to pay all the cost himself. The distribution of the burden of providing a public good in a small group is never proportional to the benefits conferred by the collective good. Olson even argues that in small groups '…there is a systematic tendency for 'exploitation' of the great by the small!'25. Selective incentives are not required in small groups; they may have the capacity to provide themselves with a collective good without relying on coercion or any positive inducement apart from the collective good itself. This is because each of the members, or at least one of them, will find that his personal gain from having the collective good exceeds the cost of providing some amount of that collective good (Olson 1965:34). In small groups free riding is not really an issue because each member, or at least some among them, is sufficiently interested in the collective good to ensure its provision. The risk of free riding is further checked by the fact that an individual's shirking behaviour would be noticeable and sanctioned.

In 'intermediate groups' no single member gets a share of the benefit sufficient to give him an incentive to provide the good himself. However, intermediate groups do not have so many members and no one member will notice whether any other is or is not helping to provide the collective good.

The likelihood that large groups are capable to provide themselves with a collective good is considered by Olson to be very small. The larger a group is, the farther it will fall short of obtaining an optimal supply of any collective good and the less it is likely to obtain even a minimal amount of such a good. In short, the larger a group, the less it will be able to further a common interest (Olson 1965:36).

Heterogeneity

The above summary of Olson's theory of how the size of the group determines collective action calls attention to an equally important variable, namely heterogeneity. In fact, Olson's definition of 'privileged group' is closely related to the assumption of groups being heterogeneous. His concept of heterogeneity, however, does not relate to socio-economic or cultural differences among group members, but to the differential value individuals place on the public good and to the resources available to contribute to its provision. The fact that generally not all individuals belonging to a group share a similar interest in a public good may appear quite obvious, but constitutes one of Olson's more important contributions and had a strong influence on further collective action research. Although homogeneous groups may exist, it is indeed misleading to treat heterogeneous groups as if they were homogeneous and to examine only the aggregate group interest in a collective good. While for Olson heterogeneity explains why some groups are more likely to succeed in collective action than others, this factor is also considered the main reason for a sub-optimal provision of the collective good. The influence of group heterogeneity on collective action has attracted many scholars and has been subject to sophisticated mathematical modelling26.

Closely related to the concept of heterogeneity is the concept of critical mass, which is used by Marwell and Oliver (1988; 1993) to explain that collective action usually depends on a relatively

[25] Olson (1965:29). Italics by the author, whereby he underlines in a footnote that "the moral overtones of the world 'exploitation' are unfortunate; no moral conclusion can follow from a purely logical analysis". The word exploitation is chosen because it is commonly used to describe situations where there is a disproportion between the benefits and the sacrifices of different people.

[26] Cf. Heckathorn (1993) and Oliver et al. (1988).

small cadre of highly interested and resourceful individuals, rather than on the efforts of the 'average' group member. These individuals behave differently from a typical group member, for example by assuming leadership positions, covering initial costs, and making up for group members who fail to contribute their share.
However, heterogeneity does not necessarily increase the probability of collective action. Hardin (1982) analyses the issue of heterogeneity in terms of 'asymmetries' that complicate the analysis of collective action. Inequality or 'asymmetries in demand for a collective good' may lead to polar opposites: to enhance or dissipate prospects for collective action. Dissipation is likely when the collective good sought by a group may also be obtained privately, although perhaps at a higher cost. In such a situation the intense demanders may opt for the private good even if it has some disadvantages. This scenario basically refers to Hirshman's (1970) distinction between 'exit' and 'voice', whereby 'exit' refers to the individual response to turn to a different product available in the market, while 'voice' is the political response to obtain a good collectively. Exit as a response to the high transaction costs of collective action, cannot be explained by focusing exclusively on group attributes but depends to a large extent on whether the context provides alternative opportunities.

<u>Interdependence</u>
Early collective action theorists, including Olson, built their theorems upon the assumption that individuals make isolated, independent decisions whether or not to contribute to collective action. Nowadays, collective action theorists recognise the critical role of interdependence between group members. Marwell et al. (1988) are among the first collective action theorists who challenge the assumption that individuals decide independently by contending that in most decision making situations, people are aware of what others are doing, as they often have social relations that make influence, or even sanctions, possible. Independent decision-making may apply to very large or 'latent' groups, but not to the more frequent, smaller groups. It may thus be assumed that people take into account whether and how much others have already contributed, and that their decisions follow a sequential pattern (Marwell et al. 1988, Melucci 1995).
Marwell et al. (1988) argue that the organisation of potential contributors, to make their decisions interdependent, is a requirement to overcome the free-rider problem. However, they do not consider the potential to organise a group to be necessarily present. The possibility of group organisation depends on the social ties in the group, particularly the density and frequency of ties, on the extent to which they are centralised in a few individuals, and on the cost of communicating and coordinating actions through these ties. The overall density of social ties in a group improves its prospects for cooperation, whereby it is noticed that network centralisation is beneficial for collective action.

The role of organisation

The role of organisation in collective action is extensively discussed by Olson. Initially, it would appear that Olson indiscriminately supports the need for organisation to pursue a common goal by arguing that:
... when a number of individuals have a common or collective interest –when they share a single purpose or objective—individual unorganised action […] will either not be able to advance that common interest at all, or will not be able to advance that interest adequately. Organisations can therefore perform a function when there are common or group interests, and though organisations often also serve purely personal, individual interests, their characteristic and primary function is to advance the common interests of groups of individuals. (Olson 1965:7)
Later in his book, however, Olson specifies that the need for organisations to pursue collective action depends on the size of the group. He does not consider the formation of organisations

necessary for small groups, in which by definition free riding and shirking would be noticeable. Moreover, given the fact that groups tend to be heterogeneous, in small groups it is likely that an individual member with a high interest in the good is willing to contribute to its provision independently of whether other group members do the same. Thus, in small groups it may not be worthwhile to bear the high transaction costs involved in establishing an organisation.
The situation changes in intermediate groups. Olson argues that for these type of groups, a collective good may or may not be obtained, but 'no collective good will ever be obtained without some group coordination or organisation' (1965:50). This is not the case in 'privileged groups', in which the good, like in small groups, may be obtained without any group organisation or coordination.
Finally, an entirely different explanation of why formal organisations are often absent and not required in relation to a particular group action is provided by the 'by-product theory'. This theory also originates from Olson (1965), who uses the concept to explain the political power of some large groups. Hence, groups may succeed in providing their collective good as a by-product of organisations based on other, selective incentives. For Olson the membership and power of large pressure-groups does not derive from their lobbying achievements, but is a by-product of their other activities. This is considered a necessary explanation for the existence -against all odds- of large voluntary organisations. The theory has the merit of calling the attention to the possibility that groups may not need to be organised because they already have been organised for other reasons.
In the present context, however, it may be sufficient to regard by-product theory as one possible explanation of why cooperation often occurs without the institutionalisation of any specific organisation. This calls attention on a critical variable that has recently been 'discovered' by collective action theorists, namely, the role of 'community'.

The role of 'community'

The discussion about the relevance of organisations, social ties and interdependence between group members leads to the 'community' concept, which has been highlighted by some authors as the key to understanding why some groups are able to solve their collective action problems without external coercion, while others are not. The importance of community for collective action was first emphasised by Taylor (1987) and further elaborated by Singleton and Taylor (1992), Taylor and Singleton (1993).
Singleton and Taylor (1992:315) define community as a set of people: (i) with some shared beliefs, including normative beliefs and preferences beyond those constituting their collective action problem; (ii) with a more-or-less stable set of members; (iii) who expect to continue interacting with one another for some time to come; and (iv) whose relations are direct (unmediated by other parties) and multiplex). They maintain that a community group is more likely to arrive at endogenous solutions to its collective action problem if it comprises individuals who are mutually vulnerable. If a group has these attributes it is more likely to be capable of facing the transaction costs related to collective action, which are the main reason why many collective action problems remain unsolved.
Axelrod (1984) emphasizes that three conditions are necessary for cooperation to be possible:

1) Individuals expected to cooperate are likely to meet again in the future;
2) Individuals must be able to identify each other;
3) They must be accountable for their actions;

People must be able to obtain information about how a person behaved in the past.
Though he does not use the term 'community' it may be noticed that these conditions are indeed fulfilled by communities as defined by Singleton and Taylor (1992). Whether virtual communities share these characteristics with real communities has been subject of lively debates

(see for example Rheingold 1993, Donath 1999 Kollock and Smith 1999) though there is a growing consensus about virtual communities functioning quite similarly to real ones (Woolgar 2000).

The importance of rules and sanctions

Whether a group is able to overcome the social dilemma of cooperation also depends on contextual factors. Indeed, in real life collective action does not occur in a socio-cultural and institutional vacuum. Whether or not people are able to pursue their common interest depends on the context and on a group's capacity to develop rules that support cooperation and discourage selfishness at the expense of common interests. According to Ostrom (1990) the sustainability of common resources depends on a number of 'design principles', such as for example clear group boundaries as to make it possible to exclude uncooperative individuals, the existence of rules that are well matched with the needs of the group, a good monitoring system and a graduated system of sanctions.

4.6.3. Application to Peer-to-Peer Networks

An application of collective action theory to the specific environment of files sharing networks available on the Internet allows us to draw some interesting conclusions that can be useful for the MobileMAN paradigm. As said, MobileMAN relies upon peer-to-peer (p2p) distributed architecture. A p2p network is a self-organizing decentralized network where its nodes both use and provide resources to other nodes. Comparison between p2p networks and MobileMAN is to be considered appropriate, since MobileMAN is a decentralized network made of nodes and without any central authority. As a consequence, the system can only work properly if users are willing to cooperate. This means that for a MobileMAN network to function cooperation either occurs on a voluntary basis or has to be enforced through positive and negative sanctions27. Understanding the behaviour of the user of p2p networks may be very valuable for the validation of the MobileMAN paradigm.

Two different p2p application types

The most common type of applications of p2p networks is "file-sharing". In this case, the users are interested in getting access to files that they do not have and downloading them. Users in general do not know each other; the user perceives other network users as archives or "libraries": mere repositories of files. Examples are Gnutella, or BitTorrent. In these applications, free-riding is very common. A second type of p2p applications is communication, such as voice-over-IP or instant messaging applications. Examples are Skype, or MSN messenger. Skype is a very interesting application, since it makes use of a strategy to allow communication between two nodes that are separated by a firewall and might be blocked to reach each other. This is based on a "bridging" mechanism that uses other Skype users to route packets to destination. For this reason, all Skype communications are encrypted, so that privacy and confidentiality are ensured. In this situation, users might be required to collaborate to enable communication between two other users. Here, however, it is less likely that users decide to logoff and close the application when not communicating with someone, because that would mean not be reachable any more. This has

[27] One example could be a function that makes it not possible to switch the device off until battery was completely consumed. In this hypothetical way, the user is forced to keep it on and provide services to other users. This example will be discussed later.

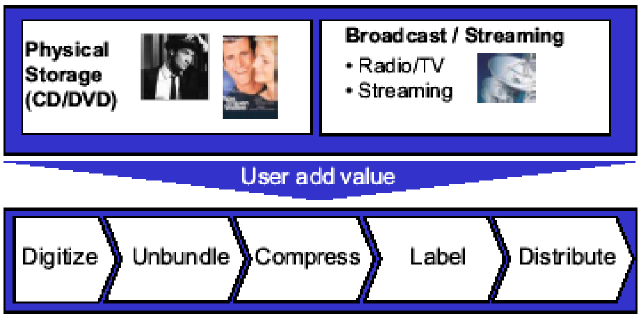
important implications for MobileMAN as it will be shown later. For now, we analyze the p2p "file-sharing" applications type.

P2p networks are collective goods shared belonging to large groups

Following Olson's group taxonomies, users of p2p networks, such as Kazaa, Gnutella, BitTorrent, can be considered as a "large group".
As was discussed earlier, according to Olson (1965), voluntary cooperation in a large group is highly unlikely. Supporting this theory is the known fact that "free-riders" make a rather large part of such p2p networks' users: Adar & Huberman (2004) found that 70% of the Gnutella network users share no files, and therefore consume resources without providing any. In fact, free riding is difficult to be checked and sanctioned accordingly. In the case of p2p, users are identified by a nickname, which can be changed at any time28, creating almost anonymous conditions. Even if a specific network might introduce positive and negative incentives, these can easily be ignored since everything happens in the virtual world. Another reason that may account for not participating can be the fact that the transaction costs are high. Users might have limited resources such as download/upload bandwidth29, disk space30 for keeping files, CPU clock time.

Nevertheless, in spite of all obstacles, p2p networks have been functioning for years and steadily growing. This means p2p networks can count on a critical mass of highly motivated individuals who ensure the network not collapsing31. A distinction must be made between "passive cooperation" and "active cooperation". According to Mannak et al. (2004), some users share because "p2p software starts sharing automatically without user interference" and "they [the users] did not feel the urge to turn off sharing". This behaviour can be labelled "passive cooperation" because it is a factual sharing but not sought and set on purpose by the user. Whereas an "active cooperation" is that by the user who actively wants to provide files and takes actions32 to do it. What is interesting is this portion of p2p networks users that do share and even more those who bear the high costs33 of providing initial files. In fact "before a user can initially supply a movie or song to a p2p network he has to digitize, unbundled, compress, and label the file" (Becker & Clement, 2004), a process that involves high costs for example in terms of time, CPU power, and work.

[28] A user can easily open a new account and behave as they were a new user – in this way, sanctions for non-cooperative behaviour can be avoided.
[29] Now this aspect could be considered as irrelevant, since Internet users have shifted from a dial-up Internet access subject to payment of per minute use to broadband DSL access with monthly fee independent from the time of use.
[30] Disk space is getting more and more important being the type of requested files movies files with an increasing quality they are quite large files that need a huge amount of space at disposal.
[31] Adar & Huberman (2004) found that 50% of the query responses – during their test of the Gnutella network – were returned by the top 1% sharing hosts. It appears that the Pareto principle governs this kind of distribution.
[32] For example, active action may be setting the preferences of the software to "share-mode" if it is not by default, moving other files in the shared folder, or provide new files (initial offer).
[33] This for example is the situation of someone who records a TV programme and converts the file in a compressed one that can be downloaded more quickly or someone who rips a DVD to share the files with others.



Value chain of user initially offering files in p2p networks.

Source: Becker & Clement (2004:2)

Another hidden cost is the risk of being caught by authorities and summoned for copyright infringement. Moreover, being media files rather large, “a user does not only internalize the costs of producing the master copy but bears opportunity costs on his hard drive as well” (Becker & Clement, 2004:2).

Small group within large group?

Why do some users not behave as typical members of large group members’ behaviour and bear the costs of sharing and of providing first instance files? It is here hypothesised that within the large group of users of a p2p network, there is a small group whose members’ behaviour is governed by a different logic34. These are the initiators of the network itself, and perhaps even the writers of the used p2p programme. These individuals know each other and belong to the same community35 sharing its own rules and norms. With this assumption, we can see that for these users being able to provide new files before others do can influence the social status within the group members. In this case, the benefit gained by bearing the costs of providing files can be perceived as higher and may trigger the continuation of sharing behaviour.

Members of this small group can be compared to the pioneers who generally are the first adopters of a new technology. Huang et al. (2004) maintain that these individuals do not need to be motivated through incentives to cooperate, because they do so voluntarily.
If the assumption that within the large group of p2p networks users there are smaller groups that are likely to bear great part of the costs of cooperative sharing behaviour, then this can explain why, despite being an anonymous large group of users, the network works and also why always greater amounts of files can be found in the distributed libraries. It is questionable whether over time with the network growing – with the number of free riders growing and the number of users cooperating remaining stable or growing as well but in a minor rate – it will not collapse because of a drop in its efficiency. Becker & Clement (2004) argue that the inevitable outcome of a growing large network of users (with a growing larger number of free riders) is its breakdown. For this reason, a later stage of the network life cycle requires cooperation enforcement mechanisms.

[34] Olson defines large groups as ‘privileged groups’ when there are some of its members who are so much interested in reaching a particular (common) goal, that they are prepared to sustain all or a great part of the costs to reach it. Maintaining this terminology, we can say that a group of p2p users is a privileged group, since a small number of its users (those of the ‘small group within the large group’) bear the costs to provide files and do not free-ride the system.
[35] Perhaps, they know each other in the real world as well. As a consequence, they are governed by the logic of small groups where each group’s member identity is defined and known to the other members.

Implications for MobileMAN

If the assumption of the existence of small groups of highly cooperative users within large groups holds true, we can expect the same situation for an ad hoc network such MobileMAN – that is, a network based on p2p principles. We can expect a highly cooperative behaviour by the pioneers and visionaries. However, as the network grows, the performance might drop dramatically, since more nodes (users) in an ad hoc network imply more traffic leading to the need to do great services in node discovery, routing, frequent update of routing tables, and so on. At this moment, it may be necessary to introduce cooperation enforcement mechanisms to ensure equilibrium between usage and service providing.

However, even to be accepted by pioneers, a new system has to present some clear advantages; advantages that sustain the mere curiosity about a new product that attracts these particular individuals that are fascinated by technology. Assuming that MobileMAN will be able to address some existing needs and that a group of pioneers decide to adopt it, at this initial stage it will not be necessary – as shown in these pages – that the system provides an incentive / cooperation enforcement mechanism, since these individuals typically present the characteristics of member of small groups whose behaviour with high probability is cooperative. At this stage, it is hypothesised that use and service providing are in equilibrium. Only at a later stage of the system adoption cycle, when the number of users grows and the group assumes the characteristics of Olson's large groups cooperation needs to be enforced. If not, the system will likely observe an unbalanced situation between use and service providing, with more use request than cooperation to service providing.

Potential Cooperation Enforcement Mechanisms for Late Stage of MobileMAN

As mentioned above, at a later stage of the MobileMAN adoption cycle a cooperation enforcement mechanism will be required. We divide the potential mechanisms in two typologies: token-based and power-based.

The first typology (token-based) relies on the concept of reputation. Reputation is a behaviour judgement that can be made visible to the others. It is the sum of different instances of behaviour of one user. It can be considered as the "social status" of someone within a particular community. Some p2p file-sharing systems make use of this concept: when requesting a file, the user receives the availability of the file as well as a queuing position. The more the user shares, the shorter the queue they have to make to download the requested file. In this way, there is a direct relationship between the reputation value and the practical advantage (convenience) of less time to wait. Reputation, however, requires that individuals are clearly and uniquely identified by other members of the group (or network). In fact, reputation as enforcement mechanism functions only if it is combined with user identification by other users. It only works if three conditions are satisfied: the user must be identified by others; the reputation value must be visible to all; the user must not be able to switch identity36. This concept comes from "the real world" – reputation in fact is one aspect that governs small groups in Olson's theory, since in small groups, the risk of free-riding is diminished because an individual non-cooperative behaviour is noticeable and sanctioned.

The second typology of incentive / sanctioning mechanisms is based on a simple idea of on/off. The system has only two states: on and off. Once turned on, it is not possible to turn it off until

[36] If users can change identity, than when someone's reputation becomes bad, then they just need to change identity and nobody will know that their behaviour with the previous identity was negative.

battery power ends. In this way, a user that switches their device on to perform a voice communication call must 'pay' for the use in battery power and has no chance to free-ride the network. It is a very simple mechanism that can be considered rather primitive. It would become more complex when inserting a third state, in between 'on' and 'off'; a state where applications are switched off but services providing functions are on. In this way, the device would only function as router for other users.
However, this second type of mechanism addresses rather delicate issues. The user must be able to decide freely if their device must be on or off. They might have the necessity to use it sporadically and the obligation to keep it on after an instance of use contradicts this right to decide. This type of cooperation enforcement mechanism is too invasive and forcing. It also opens the debate about how far it is acceptable to compel a user to cooperate: it is true that in order to work the system needs cooperation of a certain percentage of users, it is also true that someone has the right to choose whether to cooperate or not.

Closing Remarks

We applied the concepts of Olson's theory of cooperative behaviour to the particular situation of p2p networks. We hypothesised that within the group of p2p network users (that is growing and has become a "large group"), there is a smaller sub-group that is therefore governed by different logic. Members of this sub-group behave in a highly cooperative way and are responsible for initiating the network and for maintaining it balanced (by counterbalancing free-riders through providing content).
We then underlined the analogies of p2p file-sharing networks with MobileMAN. We agreed with Huang et al. on the suggestion that the first adopters of the MobileMAN technology would not need a cooperation enforcement mechanism, since their behaviour can be ascribed to the one typical of members of small groups (cooperative). Eventually, when MobileMAN would be in a later adoption stage, cooperation enforcement mechanism are likely to be necessary to prevent the majority of users from free-riding the system, that is, from using it for their needs and not cooperating for others' needs. We grouped such mechanisms in two categories: token-based and power-based mechanisms. A brief description of these two typologies was provided together with a discussion of their implications.

4.6.4. Analytical Summary and Conclusions

This review of social science theories and concepts on cooperation clearly indicates that cooperative behaviour is considered a primordial social dilemma for there is often a conflict between individual and group interests. Nevertheless theory and research indicate that though cooperative behaviour should not be taken for granted, people are more likely to pursue group interests if some conditions are fulfilled. These conditions were extensively discussed and primarily refer to group attributes (size, heterogeneity and interdependence), on whether the group may be considered a 'community' whose member interact with each other on a sustained basis, and on other contextual factors. Further, a group's capacity to develop its own rules and regulations, to provide incentives for cooperation and to punish selfish behaviour are some of the additional prerequisites to pursue collective interests.
Social science literature thus confirms what was pointed out by Huang et al. (2004), namely that mobile metropolitan networks which depend on the cooperation of individual users may be viable and effective for small groups, but may face serious constraints in case of large groups of unrelated users. Indeed, whereas in small groups non-cooperative behaviour may be easily detected and sanctioned, this becomes technically and organisationally far more complex in large groups. If cooperation in large ad hoc networks leads to excessively high individual costs or burdens it may only be achieved if the system entails some 'selective incentives' that allow to

reward cooperative behaviour and sanction selfishness. From a technical point of view this apparently still poses a number of problems (Huang et al. 2004)

It was discussed that one of the reasons why cooperation often does not occur is because the so-called transaction costs. As pointed out by Rheingold (2002) and Mele (1999) many ICTs significantly contribute to reduce the costs of cooperation and thus gave a strong impetus to social movements and collective action. However, whether individuals are willing to bear the transaction cost associated with sharing a collective good depends to a large extent on the existence of alternative products. Indeed, as pointed out by Hirshman already in the 1970s, people are likely to 'exit' from a situation that requires cooperation if they can obtain the same service from a source that does not have that requirement, as to avoid risk, uncertainty and dependence. Thus, only if MobileMAN will offer some unique applications, which cannot be obtained from other competing communication technologies that do not require cooperation, will users be motivated to overcome the social dilemma of cooperation.

To conclude, it may be premature at this stage of the development of the MobileMAN to make strong statements about whether cooperation will be a major constraint for the social viability of this paradigm. This will indeed primarily depend on the cost of cooperative behaviour (e.g. battery power) and on the type of applications, i.e. whether MobileMAN will have some highly valued services to offer that cannot be obtained from other technologies. If –as argued by Huang et al. (2004) – the cooperation costs in ad hoc networks are likely to be minor there will be no good reason for individuals to shirk and to act selfishly. In any case, the fact that communication in ad hoc networks may be free of charge will only be considered an advantage if the system is reliable and the transaction costs are kept low.

4.7. References

Ackermann, T. et al. (2002) Incentives in Peer-to-Peer and Grid Networking. UCL Research Note RN/02/24.2002.

URL: http://www.cs.ucl.ac.uk/staff/T.Ackemann/papers/incentives.pdf

Adar, Eytan & Huberman, Bernardo (2000) Free Riding on Gnutella. First Monday Vol. 5, Number 10, - Oct. 2nd 2000.

Anagnostakis, Kostas & Greenwald, Michael (2004) Exchange-based Incentive Mechanisms for Peer-to-Peer File Sharing.

URL: http://www.cis.upenn.edu/~anagnost/papers/exchange.pdf

Axelrod, R. 1984. The Evolution of Cooperation. New York: Basic Books.

Becker, Jan & Clement, Michel (2004) The Economic Rationale of Offering Media Files in Peer-to-Peer Networks. Proceedings of the 37th Hawaii International Conference on System Science. URL:http://csdl.computer.org/comp/proceedings/hicss/2004/2056/07/205670199b.pdf

Felkins, Leon (2000) The Voter's Paradox.
URL: http://www.magnolia.net/~leonf/sd/vp-brf.html

Gurak, L. J. 1999. The Promise and the Perils of Social Action in the Cyberspace: *Ethos*, Delivery and the Protests over the MarketPlace and the Clipper Chip. In: M. Smith and P. Kollock, *op. cit.*

Hardin, G. 1968. The Tragedy of the Commons. *Science* 162: 1243-48.

Hardin, R. 1982. Collective Action. Baltimore, John Hopkins UP.

Heckathorn, D. D. 1993. Collective Action and Group Heterogeneity: Voluntary Provision Versus Selective Incentives. *American Sociological Review*, Vol 58: 329-350.

Huang, E., J. Crowcroft and I. Wessel. 2004. Rethinking Incentives for Mobile Ad Hoc Networks. Paper presented at the ACM SIGCOMM Conference. Portland, Oregon. September 2004.

Huberman, Bernardo & Lukose, Rajan (1997) Social Dilemmas and Internet Congestion.
URL:
http://www.hpl.hp.com/research/idl/papers/InternetCongestion/InternetCongestion.pdf
Kollock, P. 1993. Cooperation in an Uncertain World: An Experimental Study. *Sociological Theory and Methods* 8/1: 3-18.

-------------. 1998a. Design Principles for Online Communities. In: The Internet and Society. URL: www.sscnet.ucla.edu/soc/faculty/kollock/papers/design.htm)

Kollock, P. 1998b. Social Dilemmas. The Anatomy of Cooperation. *Annual Review of Sociology 24*: 183-214.

--------------. 1999. The economies of online cooperation: gifts and public goods in the cyperspace. In: M. Smith and P. Kollock, *op. cit.*

Kollock, Peter & Smith, Marc (1996) Managing the Virtual Commons: Cooperation and Conflict in Computer Communities. In Computer-Mediated Communication: Linguistic, Social, and Cross-Cultural Perspectives. Ed. Herring, S. Amsterdam: John Benjamins.
URL: http://www.sscnet.ucla.edu/soc/faculty/kollock/papers/vcommons.htm

Hirschman, A.O. 1970. Exit, Voice and Loyalty. Responses to Decline in Firms, Organizations and States. Cambridge (Massachusetts), Harvard UP.

Mannak, R. et al (2004) Understanding Sharing in Peer-to-Peer Networks.
URL: http://www.cactus.tudelft.nl/CactusPublications/understandingsharing%20FINAL.pdf
Marwell, G. and P. E. Oliver. 1988. Social Networks and Collective Action. A Theory of the Critical Mass. *American Journal of Sociology* 94/3: 502-534.

Marwell, G. and P. Oliver. 1993. The Critical Mass in Collective Action. A Micro-Social Theory. Studies in Rationality and Social Change, Cambridge University Press.

Mele, C. 1999. Cyperspace and Disadvantaged Communities: the Internet as a tool for Collective Action. In: M. Smith and P. Kollock, *op. cit.*

Melucci, A. 1995. Challenging Codes. Collective Action in the Information Age.

Nagel, J.H. 1987. Participation. Englewood Cliffs (New Jersey), Prentice-Hall. North, D.C. 1990. Institutions, Institutional Change and Economic Performance. Cambridge University Press.

Olson, M. 1965. The Logic of Collective Action. Cambridge, Mass., Harward UP. 206)

Ostrom, E. 1990. Governing the Commons. The Evolution of Institutions for Collective Action. New York: Cambridge UP.

Rheingold, H. 1993. The Virtual Community. Homesteading on the Electronic Frontier. Reading, MA, Addison-Wesley

----------------. 2002. Smart Mobs. The Next Social Revolution. Boulder (CO): Perseus Press.

Runge.C.F. 1984. Institutions and the Free Rider: The Assurance Problem in Collective Action. *Journal of Politics* 46:154-81.

--------------. 1992. Common Property and Collective Action in Economic Development. In: Bromley (ed.), *op. cit.*

Sandvine Inc. White Paper (2003) Regional characteristics of P2P. File sharing as a multi-application, multi-national phenomenon.
URL: http://www.slyck.com/misc/Euro_Filesharing_DiffUnique.pdf
Singleton, S. and Taylor, M. 1992. Common Property, Collective Action and Community. *Journal of Theoretical Politics* 4(3):309-324.

Smith, M. A. 1999. Invisible Crowds and the Cyberspace: Mapping the Social Structure of the Usenet. In: M. Smith and P. Kollock, *op. cit.*

Taylor, M. 1987. The Possibility of Cooperation. Cambridge University Press.

------------------. 1988. Village Republics. Economic Conditions for Collective Action in South India. Cambridge University Press, Cambridge.

Wellman, B. and M. Giulia. 1999. Virtual communities: Net Surfers don't Ride Alone. In: M. Smith and P. Kollock, *op. cit.*

Woolgar, Steve (ed.) 2000. Virtual society? Technology, cyberbole, reality. Oxford: Oxford University Press.

[HJP02]	Y. Hu, D. B. Johnson, and A. Perrig, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings of MobiCom '02*, Atlanta, Georgia, USA, Sept. 2002.
[SGF02]	R. Schollmeier, I. Gruber, and M. Finkenzeller, "Routing in Mobile ad Hoc and

	Peer-to-Peer networks. A Comparison," in *Proceedings of Networking 2002 Workshops*, Pisa, Italy, May 2002.
[PC02]	I. Pratt and J. Crowcroft, "Peer-to-Peer systems: Architectures and Performance," Pisa, Italy, May 2002.
[AH00]	E. Adar and B. Huberman, "Free riding on gnutella," *First Monday*, vol. 5, no. 10, Oct. 2000.
[SGG02]	S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, Jan. 2002.
[MM03]	P. Michiardi and R. Molva, "Security in Ad Hoc Networks," in *Proceedings of the Eighth International Conference on Personal Wireless Communications (PWC 2003)*, Venice, Italy, Sept. 2003, pp. 756–775.
[SP03]	J. Shneidman and D. Parkes, "Rationality and self-interest in peer to peer networks," in *Proceedings of the 2nd Int. Workshop on Peer-to- Peer Systems (IPTPS'03)*, Berkeley, CA, USA, 2003.
[H68]	G. Hardin, "The Tragedy of the Commons," *Science*, no. 162, pp. 1243–1248, 1968.
[GLML01]	P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge, "Incentives for Sharing in Peer-to-Peer Networks," in *Proceedings of the Third ACM Conference on Electronic Commerce*, Tampa, Florida, USA, Oct. 2001.
[KYG03]	S. Kamvar, B. Yang, and H. Garcia-Molina, "Addressing the Non-Cooperation Problem in Competitive P2P Systems," in *Proceedings of the First Workshop on Economics of P2P Systems*, June 2003.
[NW03]	P. D. T.W. Ngan, D. S. Wallach, "Enforcing Fair Sharing of Peer-to-Peer Resources," in *Proceedings of the 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, USA, 2003.
[IEEE99]	IEEE, "Standard for Wireless LAN-Medium Access Control and Physical Specification, P801.11," *Part 11*, 1999.
[KV03]	P. Kyasanur and N. Vaidya, "Detection and Handling of MAC Layer Misbehavior in Wireless Networks," in *Proceeding of Dependable Computing and Communications Symposium (DCC) at the International Conference on Dependable Systems and Networks (DSN)*, June 2003.
[RT99]	E. Royer and C.-K. Toh, "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks," *IEEE Personal Communications*, Apr. 1999.
[MM02]	P. Michiardi and R. Molva, "Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks," in *Proceedings of European Wireless 2002 Conference*, Feb. 2002.
[SCWA99]	S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP Congestion Control with a Misbehaving Receiver," *ACM Computer Communications Review*, pp. 71–78, Oct. 1999.
[CRVP01]	K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback Based Scheme for Improving TCP Performance in Ad Hoc Networks," *IEEE Personal Communication Systems Magazine: special issue on Ad Hoc Networks*, vol. 8, no. 1, pp. 34–39, Feb. 2001.
[HV99]	G. Holland and N. H. Vaidya, "Analysis of TCP Performance Over Mobile Ad Hoc Networks," in *Proceedings of IEEE/ACM MOBICOM '99*, Seattle, Washington, Aug. 1999.
[LS01]	J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, July 2001.
[BH03]	L. Buttyan and J. Hubaux, "Stimulating cooperation in self-organizing mobile ad

	hoc networks," *ACMlkluwer Mobile Networks and Applications (MONET)*, vol. 8, no. 5, Oct. 2003.
[ZCY03]	S. Zhong, J. Chen, and Y. Yang, "Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks," in *Proceedings of IEEE Infocom '03*, San Francisco, Mar. 2003.
[CG03]	J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring, "Modelling Incentives for Collaboration in Mobile Ad Hoc Networks," in *Proceedings of WiOpt '03*, Sophia-Antipolis, Mar. 2003.
[MGLB00]	S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," in *Proceedings of the Sixth annual ACM/IEEE International Conference on Mobile Computing and Networking*, 2000, pp. 255–265.
[BL02]	S. Buchegger and J. Y. L. Boudec, "Performance analysis of the CONFIDANT protocol," in *Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, June 2002.
[MM02s]	P. Michiardi and R. Molva, "CORE: A COllaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks," in *Proceedings of Communication and Multimedia Security 2002 Conference*, Sept. 2002.
[BAS03]	C. Buragohain, D. Agrawal, and S. Suri, "A Game Theoretic Framework for Incentives in P2P Systems," in *Proceedings of the Third IEEE Conference on Peer-to-Peer computing*, Sept. 2003.
[K98]	P. Kollock, "Social Dilemmas: The Anatomy of Cooperation," *Annual Review of Sociology*, no. 24, pp. 183–214, 1998.
[K03]	S. Kuhn, "Prisoner's Dilemma," in *Stanford Encyclopedia of Philosophy*, E. N. Zola, Ed., 2003, pp. 183–214.
[GU04]	S. Giordano and A. Urpi, "Self-Organized and Cooperative Ad Hoc Networking," in *Mobile Ad Hoc Networking*, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, Eds. IEEE Press and John Wiley and Sons, Inc., New York, 2004.
[SNCR03]	V. Srinivasan, P. Nuggehalli, C. Chiasserini, and R. Rao, "Cooperation in wireless ad hoc networks," in *Proceedings of IEEE Infocom '03*, San Francisco, Mar. 2003.
[FBH03]	M. Felegyhazi, L. Buttyan, and J.-P. Hubaux, "Equilibrium Analysis of Packet Forwarding Strategies in Wireless Ad Hoc Networks – The Static Case," in *Proceedings of the Eighth International Conference on Personal Wireless Communications (PWC 2003)*, Venice, Italy, Sept. 2003.

5. MOBILEMAN MIDDLEWARE ARCHITECTURE

5.1. Introduction

With the term *ad hoc*, we refer to the art of networking without an infrastructure. The nodes of an ad hoc network are mobile, and self-organize to provide and use services, guaranteeing wireless communication to end-users.

The distributed nature of ad hoc networks fits well the peer-to-peer (p2p) model of computation. P2p systems are those where centralization is not possible, relations are transient, and resources are highly distributed. These are also main requirements in ad hoc environments. This computational duality suggests taking into account the p2p systems state of the art, and that ad hoc networking can learn important lessons from what has been proposed for the Internet.

In the context of this work, we focus on a p2p middleware platform called Pastry [RD01]. This platform exports a *subject-based* data routing interface, working on top of an overlay network. With the term subject-based, we refer to the ability of routing data items in a distributed virtual space using data subjects (e.g. the filename of a file, the identifier of an object etc.). This technique works both to fill up and query the distributed virtual space, provided that data producers and consumers agree on the subjects. Interesting services and applications [FreePastry] have been built over Pastry, showing that this interface is general enough to be used also in ad hoc environments.

A platform like Pastry concretely helps the development of resilient and fully distributed systems, where the overall workload is fairly spread among the systems participants. These are fundamental characteristics in ad hoc contexts. Resiliency is mainly needed because of nodes mobility, as well as the fact that users may turn services (and eventually devices) on and off at their pleasure. Fair workload sharing would be desirable to avoid central points of failure, and situations with congested nodes. Moreover, the data routing policy guarantees upper-bounds on the number of forwarding hops required to insert or retrieve a data item in the system. However, Pastry has been designed for large networks (i.e. the Internet), where thousands of nodes wish to participate to the p2p service. Subject-based routing results efficient as each node builds "enough" knowledge about the rest of the participants (the overlay network), and maintains it coherent with changes. This is costly, and even if ad hoc networks will be much smaller than thousands of nodes, an out-of-the-box Pastry platform would have to cope with highly variable network topologies, where many nodes arrive/exit and connections frequently result intermittent.

This document introduces innovative protocol stack architecture for ad hoc networks, where emphasis is given to *cross-layering*. The key idea is that the information collected by each protocol could be used inside the stack to optimize tasks of other protocols. The focus in this work is on the cross-layer interaction between a proactive routing protocol and a subject-based routing substrate like Pastry. We give perspectives on how costs and complexity of building and maintaining a Pastry overlay network can be reduced through cross-layer interactions.

5.2. Overview of Pastry

Pastry [RD01] supports the development of fully distributed p2p systems, providing a subject-based data routing interface. Its data routing policy converges in a bounded number of forwarding hops in the overlay network. This is done using a small per-node routing table.

Pastry works with a circular address space of size 2^{128}-1, called *ring* of addresses. A node becomes a peer of a Pastry network, getting an address from the relative ring. This is done hashing, for example, its IP address or its public certificate. Peers of a Pastry ring distribute *(key, value)* pairs among themselves, hashing key components and retrieving their logical position in the ring. The distribution strategy maps a key *k* on the peer that has the logical address numerically closer to *k*. This is done considering node identifiers and keys as sequences of digits

with base 2^b. Pastry routes messages hop by hop, getting each time closer to the target node. At each hop, a peer P1 forwards the packet to a peer P2 sharing with *k* a prefix that is at least one digit (or *b* bits) longer than P1 logic address. If no such peer is known, the message is forwarded to the node whose identifier shares the same prefix as P1, but is numerically closer to *k*.
To route messages each peer builds and maintains three data structures. The first one is a routing table *R*, organized into $\log_{2^b} N$ rows (N is the size of the p2p network) each with 2^b - 1 entries. Each entry at row *n* refers to a peer whose identifier has a common prefix of length *n* with the current peer, but differs at digit *n + 1* that is the same of the entry index. Each entry in *R* contains the logical address of one of (potentially) many nodes taking part to the ring. In case of multiple nodes, the choice may be driven by proximity metric (e.g. the distance in number of hops from the present node), in order to guarantee good locality properties. The entry is empty if there are no suitable nodes. The second structure is the neighborhood set *M*, which contains references to other peers that are close (according the proximity metric) to current one (physical neighbors). *M* is not used in message routing, but it is used to maintain locality properties and periodically monitor the overlay structure. Finally, a leaf set *L* maintains references to peers that are logically close (i.e. in the Pastry ring) to the current node. *L* is centered around the current peer identifier, and is used during the message routing in the following way: in forwarding a given message, the peer first checks if the key falls within the range of identifiers covered by its leaf set and in this case it sends the message directly to the destination, otherwise it applies the Pastry prefix-based routing looking for the best match into *R,* probably starting a multi-hop routing of the message.
When a new node *X* arrives, it needs to initialize its state structures, informing other nodes about its presence. *X* has to first contact another nearby peer A, according to the proximity metric, asking it to route a message with key equal to X. This message will be eventually delivered to the peer Z logically close to X. All the intermediate peers in the routing path from A to Z, will send their structures to X. X will build its state synthesizing the received structures in the following way: the neighborhood set M from A, the leaf set L from Z, and the i-th row of the routing table R from B_i, which is the i-th peer in the path from A to Z and shares a prefix of length i with X. Finally, X informs the previous peers about its arrival, transmitting a copy of the synthesized state. In order to maintain the overlay structure it is also necessary that each peer executes a polling procedure toward its physical neighbors in order to discover their status and possible disconnection events. These procedures ensure that the overlay is built and maintained correctly.
There are other proposed systems implementing subject-based routing on top of structured p2p overlays. CAN [RFHKS01], Chord [SMKKB01], Tapestry [ZKJ01] and Kademlia [MM02] build the overlays differently, but in principle they all could export the same interface to programmers, as proposed in [DZDKS03].

5.3. Pastry in the cross-layer architecture

Pastry prefix-based routing converges in a logarithmic number of steps respect to the overlay size, provided that data structures such as routing tables and leaf sets, are well-formed. Typically, Internet nodes have little knowledge about the network topology: apart from routers, which do not participate to Pastry rings; normal nodes only know their subnet gateway to forward locally generated packets. This implies that in order to be part of a Pastry overlay network, where packet forwarding on behalf of others is required, peers have to gather knowledge about the overlay topology. Building and maintaining coherent Pastry structures is costly. In [RD01] the authors performed a cost analysis for a network of 5000 peers, where a 10% failure rate affects randomly selected nodes. As a result, each node made an average of 57 remote procedure calls just to repair the tables. Ad hoc environments will put a Pastry overlay through tougher conditions. For example, nodes mobility and users turning on and off their devices will cause peers to be intermittent, and entire rings to frequently split and rejoin. Even if ad hoc networks are smaller

than thousand of nodes, their dynamics could quickly bring an "out of the box" Pastry to unsatisfactory performances.
Hereafter we analyze how to reduce the cost associated with structured overlay management, when the underlying physical network is ad hoc and the cross-layer architecture presented in Section 1 is adopted. In this case, mobile nodes have resource limitations can be balanced by the exploitation of nodes knowledge about the network topology, because they actively participate to routing. If such knowledge is made available in the stack, for example through the cross-layer architecture previously described, then other protocols could benefit from it optimizing their functioning. In the case of Pastry, accessing the routing tables through the NeSt would mean to have a locally available references to a set of nodes (those forming the physical network), potentially including the whole overlay. This suggests that a great amount of p2p communication to gather overlay knowledge from other peers can be saved.

5.4. Discovering overlay peers in the physical network

The Link-State protocol continuously updates a representation of the network topology as it receives LSUs coming from other nodes. The NeSt reflects changes performed in the routing tables, allowing other protocols in the stack to read the updated topology representation. A node running Pastry and willing to join a previously established ring R in the network, would need to understand which nodes, among those visible through the NeSt, are currently making R up. As Pastry rings are application layer services, this is equivalent to say that the joining peer needs a *service discovery module*. Currently, a joining peer either has a reference to neighbor peer already in R, or bootstraps a newly created ring. This scenario is realistic in networks with infrastructure, but does not apply in self-organizing ad hoc networks.
Our solution is based on the use of the Service Discovery Module (SDM), presented in Section 3.3, which disseminates small *service extension* blocks in the LSU packets.[37] On each node, an SDM module (see Figure 5.1) collects incoming service extension blocks, locally building a network service map. The overhead introduced by this cross-layer interaction is well compensated by the elimination of an explicit service discovery protocol.

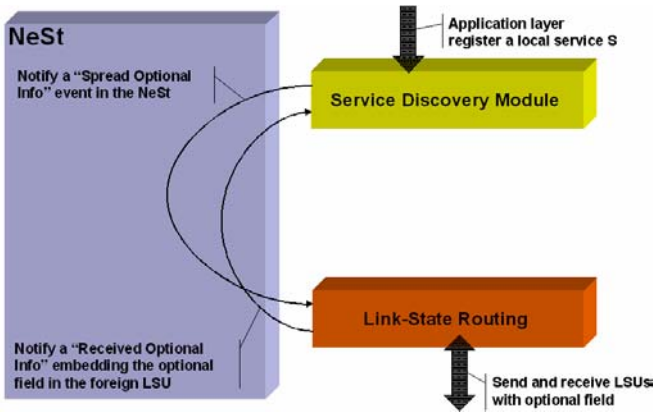

NeSt
Notify a "Spread Optional Info" event in the NeSt
Application layer register a local service S
Service Discovery Module
Link-State Routing
Notify a "Received Optional Info" embedding the optional field in the foreign LSU
Send and receive LSUs with optional field

Figure 5.1: Cross-Layer interaction between routing and SDM.

Figure 5.1 shows the cross-layer interaction between the SDM and the link-state routing. Moreover, the SDM exports an interface to interact directly with middleware platforms (e.g. Pastry) and applications.

37 By broadcasting few bytes with LSU we obtain a proactive service discovery, without the need for explicit service discovery communications.

A Pastry instance runs to support a ring R (i.e. a distributed p2p service), and can register basic information identifying R to the SDM. Additionally, Pastry can also query the SDM to get the addresses of those nodes running the same ring R.
The SDM collects all the locally registered services, exporting them to the NeSt in the form of service extensions for LSUs. It also registers to the NeSt a subscription for service extensions generated by other nodes, received through incoming LSUs.
The link-state routing reads local service extensions from the NeSt, piggybacking them inside newly produced LSUs. As nodes participate to the link-state routing, enhanced LSUs travel the network, reaching far away nodes. Each received LSU is parsed out to get link-state information useful for building/updating routing tables, and eventual service extensions to be notified to the NeSt.
The NeSt supports and coordinates cross-layer interactions, handling shared data and events subscriptions and notifications.
In the following section advantages of this new middleware platform for ad hoc networks based on the cross-layer architecture will be detailed.

5.5. CrossROAD: Pastry & Cross Layering

Applying Pastry [RD01] through classical legacy architecture to ad hoc networks is not the real solution to realize an optimized middleware platform for this kind of scenario. In fact Pastry is designed for wired network where thousands of nodes take part to the same service in order to share and exchange information. In this case, nodes have generally fixed positions and they have not power constraints or connection problems, while in ad hoc networks mobile nodes cause frequent topology updates due to their movements and also to possible coverage problems. In addition they also have to save energy correctly managing their resources.
The classical Pastry model does not care of these aspects and particularly defines overlay management policies, described in the previous section that can negatively influence ad hoc network performances. Particularly, join operations and monitoring overlay status require a lot of remote connections, not only to physical neighbors, producing a big overhead for ad hoc networks. In addition the distribution and recovery of information, forcing the message forwarding to use additional path due to the subject-based policy, can introduce a further overhead.
Applying the cross-layer architecture in order to share information between the network layer and the middleware layer, we have planned a new solution that has been named *CrossROAD*: *CROSS-layer Ring Overlay for AD hoc networks* [Del05].
In the following subsections we will detail CrossROAD design pointing out main advantages of this solution.

5.5.1. Node Identifiers and logical address space

As already described in the previous section, Pastry assigns a 128-bit logical address to each node willing to join a ring. This address will represent it in the ring. This is typically done hashing the IP address, the hostname or the public key of the local node, so that those nodes that are physical neighbors are logically scattered on the ring. The hash function used to convert physical addresses in logical addresses uniformly and randomly distributes the logic addresses on the ring, thus guaranteeing a fair balancing of the amount of keys each node will be responsible for.
The same principle is applied in CrossROAD, but in this case, using cross-layer architecture, each node can directly know the entire network topology from the routing table managed by the network routing protocol, based on a proactive approach. Since the network routing table contains the IP address of each node of the network, the node identifier can be represented by the hash function applied to this information, such that the local node can directly build the overlay network simply applying the hash function to the IP address of the nodes providing the same

service. Hence, when a node decide to join a ring, CrossROAD first has to know which other nodes in the network take part to the same ring , and to do this it can directly access to the NeSt where those information are stored. Once CrossROAD knows the other participants, it has only to locally compute their logical addresses and store this information in the local data structures. In this way, all remote connections required by Pastry for each join operation are removed and there is no additional overhead to build the ring overlay.

5.5.2. Managing overlay data structures

In order to manage the overlay data structures consequently to join and disconnection events, Pastry requires several remote connections, increasing the overhead of ad hoc network communications. In addition it has to manage three different data structures to maintain the correspondence between the physical and the logical topology. Particularly it stores the node identifiers of strict logical neighbors in the *leaf set*, those of physical neighbors in the *neighborhood set* and the others sharing a common prefix with the logical address of the local node in the *routing table*.
CrossROAD avoids all remote connections for the overlay management exploiting cross-layer architecture functionalities. In addition it reduces the data structures used to define the overlay to a simple routing table while leaf set and neighborhood set disappear. In fact the routing table contains the logical addresses of all nodes taking part to the same service, organized following the prefix-based metric in order to maintain the subject-based routing principle. In this way the routing table size depends on the number of nodes providing the service and, in the worst case, it is equal to the network size if all nodes participate to the ring. At the same time each node has a complete knowledge of the overlay network. Once the CrossROAD routing table is locally built, consequently to the join operation of the local node, it also has to be maintained accordingly to the physical network topology and to additional connection and disconnection events. To this aim, Pastry defines a polling procedure limited to the physical neighbors, in order to discover their status. In fact a remote node is considered disconnected from the Pastry network if it doesn't answer to a polling message before timeout expiration. In this way Pastry requires additional remote connections that negatively influence network performances.
Even in this case CrossROAD does not require any remote connection thanks to the network routing protocol that, periodically sending its LSU packets, recovers all the topology updates and directly provides to update the routing table and its abstraction in the NeSt. In this way CrossROAD, each time it has to send a message on the overlay network, it has to verify in the NeSt if the content of its routing table is coherent with the current topology of the network, otherwise it has to update it before sending the message.
With this solution the overlay management is enormously simplified and there is no additional overhead to the ad hoc network functionalities.

5.5.3. CrossROAD Subject-based routing

The main characteristic of the structured overlay model based on Pastry is represented by the subject-based routing defined to distribute and recover data on the network. Since the overlay data structures in Pastry has a fixed dimension depending on the network size, they cannot maintain the entire set of nodes taking part to the ring and, for this reason, the subject-based routing is represented by a multi-hop routing at the middleware level if the selected destination is not part of the logical neighbors of the sender (see Figure 5.2). This implies that at the network layer, the message forwarding is forced to send the message to intermediate nodes, logically nearer to the key of the message, extending the optimum path between the source and the destination. This creates an additional overhead to the ad hoc network communications that is eliminated by CrossROAD, thanks to the complete knowledge of the overlay. In fact, since each node knows the others, the sender can directly recover the nearest destination for a selected key and directly send the message through a simple peer-to-peer connection, while the forwarding protocol at the network layer provides to deliver the message through the shortest path (see Figure 5.3). In this way the logarithmic lookup cost depending on the network size is further reduced in CrossROAD to a constant value, independently from the network dimension.

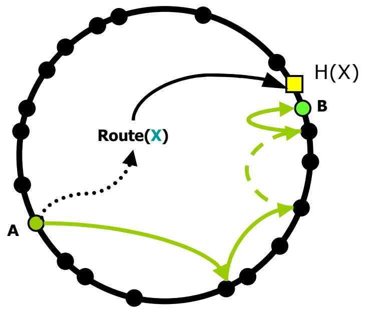


Figure 5.2: Routing a message between nodes in Pastry

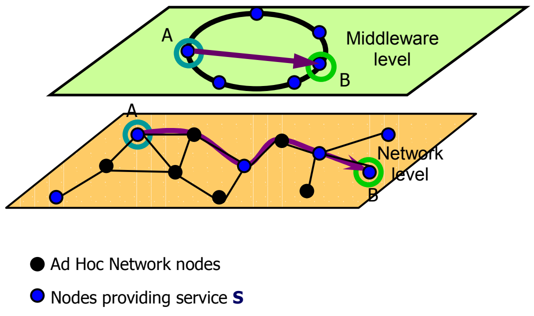


Figure 5.3: Routing a message in CrossROAD and related network forwarding

5.6. CrossROAD Software Architecture

In order to obtain a complete view of the system architecture, it is not possible to limit this paragraph to the software architecture of CrossROAD. Since it is based on a cross-layer interaction with a proactive routing protocol, its software architecture is strictly connected to the NeSt architecture. As shown in Figure 5.4, the NeSt architecture can be limited to two main packages aimed at managing services information, directly connected to CrossROAD through a simple interprocess communication. Every time a new instance of CrossROAD is created, because a new service starts running upon it, it sends a request of publishing the related service to the NeSt. Then it will forward this information to the routing protocol in order to broadcast it on the network. In addition, when CrossROAD has to send an application message on the overlay, it requires to the NeSt the current state of the network topology and the list of nodes currently providing that specific service. To do this, the proactive routing protocol has to send to the NeSt an abstraction of the network topology with the same frequency with which it updates its internal data structures.

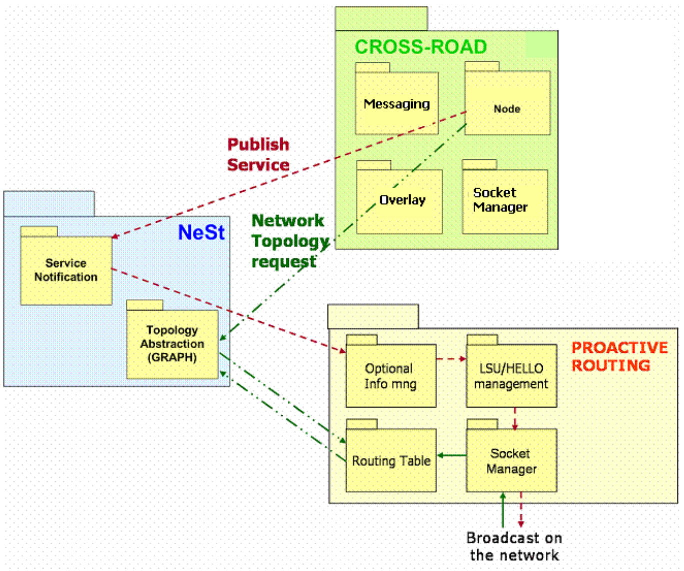


Figure 5.4: Overview of the software architecture

For the development of a MobileMAN prototype, we choose to limit the implementation of the cross-layer architecture to middleware and routing interactions exploiting an open source implementation of the proactive routing protocol OLSR [Uni] (see Section 1.5). This implementation allows the development of dynamic libraries (plugins) for the definition of additional information to be sent on the network through routing packets. In case of CrossROAD, this library has been called *XL-plugin*, because it implements cross-.layer interactions between middleware and routing protocols.

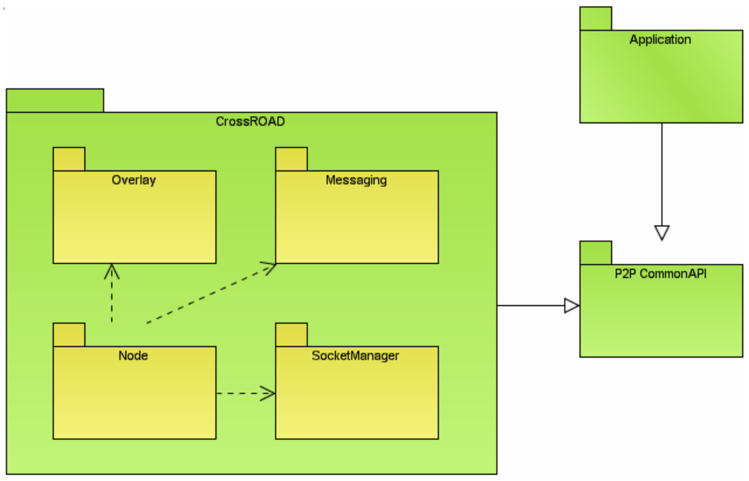


Figure 5.5: CrossROAD package diagram

Before explaining how CrossROAD interacts with the XL-plugin to create and manage the overlay, it is important to describe its complete software architecture. As shown in Figure 5, CrossROAD consists of four main packages aimed at implementing different functionalities and managing related data structures. Specifically, we can describe them as follows:

- *Overlay:* it contains all data structures designed to collect the overlay routing table and the association between each node identifier and its IP address;

- *Messaging:* it defines all messages used by CrossROAD to communicate with the application and the XL-plugin;
- *SocketManager:* it manages all local and remote sockets used to communicate with the application, the XL-plugin, and other instances of CrossROAD running on different nodes for the same service.
- *Node:* it represents the kernel of this middleware platform. Not only it defines all objects necessary to interact with other packages, but it also generates all threads designed for the management of the overlay, for the establishment of remote connections towards remote nodes and for the maintenance of the cross-layer interaction with the routing protocol through the XL-plugin. It is also in charge of correctly managing disconnection and failure events.

It is important to notice that CrossROAD implements the P2P CommonAPI defined in [DZDKS03] maintaining the application completely transparent respect to the creation of specific instances of CrossROAD objects. This API has been designed to allow each distributed application that implements it, to run on middleware platforms providing it. Specifically, to better understand how a distributed application has to be defined in order to run on top of CrossROAD; a detailed description of the CommonAPI is given in Figure 5.6.

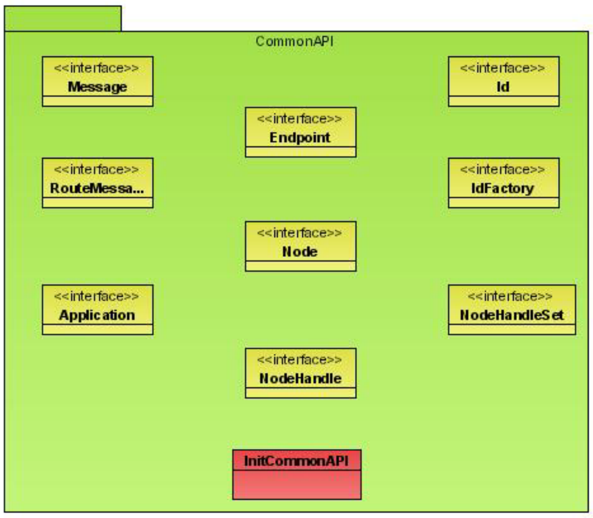


Figure 5.6: P2P CommonAPI class diagram

The CommonAPI package consists of 9 interfaces and one class aimed at initializing them with the related instances of CrossROAD objects (*InitCommonAPI*). The most part of these interfaces are implemented by CrossROAD, while others must be implemented by each specific application:

- *Id:* it represents an abstraction of logical identifiers, defining methods that allow the application to recover this information. It is implemented by CrossROAD defining how each identifier has to be represented (160-bit).
- *IdFactory:* it represents the abstraction of the method chosen to calculate each logical identifier. In case of CrossROAD, it is implemented as a SHA-1 hash function.
- *Message:* abstraction of all messages that have to be sent on the network through the overlay. In order to allow the correct transmission of all messages, the message data structures of each application have to implement this interface.
- *RouteMessage:* abstraction of messages that have to be forwarded to another node of the overlay. It is implemented by CrossROAD to distinguish application messages

from middleware messages that are enhanced with the key value for the subject-based routing.
- *Node:* abstraction of the main class of each overlay network. Every middleware instance is identified by an entity that is in charge of managing all overlay data structures and every thread related to different sets of functionalities of the entire system. It is implemented by CrossROAD.
- *NodeHandle:* abstraction of the couple (node identifier, IP address) necessary to maintain a correspondence between physical and logical address spaces. It is implemented by CrossROAD.
- *NodeHandleSet:* abstraction of a set of NodeHandle, used to select a group of possible destinations for each single message. It is used in case of data replication management. It is implemented by CrossROAD.
- *Endpoint:* abstraction of the entity that connects the instance of the local Node to the related application. It is implemented by CrossROAD to define processing operations of application messages before sending them on the overlay. When the application wants to send a message with a specified key value, it has to call the function `“`**`route`**`(Id id, Message message, NodeHandle hint)”`, where `id` represents the key value, `message` the application message, and `hint` represents a destination for the message specified by the application, bypassing the subject-based routing. For the normal use of CrossROAD, it is set to null value, except in case of particular actions from the application.
- *Application:* abstraction of each distributed application developed to run on top of a structured overlay network. It defines three main functions to be implemented by each application with the following purposes: forwarding an application message on the overlay, receiving an application message form another node running the overlay and notifying join and disjoin operations of the local node to the overlay data structures.

Specifically, these functions have the following headers:

- `public void `**`deliver`**`(Id id, Message message)`
  This method is called by the middleware when the overlay receives a message with a key value logically closest to the local node identifier. It has to be implemented by the application to process the content of the message and eventually update its local data structure.

- `public boolean `**`forward`**`(RouteMessage message)`
  This method is invoked on the application when the underlying node is about to forward the given message with a specific target to another node passing through the local one. It is called also on the key’s root node (before deliver is invoked). In this case the application could change the contents of the message, specify a different destination, or completely terminate the message. In case of changes, the function returns true and the node has to check the final destination of the message and it forward it without invoking the deliver function on the local node. This function has been preserved from the original definition of the P2P CommonAPI even if in case of CrossROAD there is no multi-hop routing at the middleware layer, but it directly connects to the best destination of a message. For this reason it is used by the application only to manipulate the message or the selected destination.

- `public void `**`update`**`( NodeHandle handle, boolean joined)`

Originally this method was designed to inform the application that the specified node has either joined or left the overlay. In case of CrossROAD, the local node is not notified every time a join or disjoin event occurs because of the high dynamicity of ad hoc networks topology. In fact a CrossROAD node becomes aware of an overlay change every time it has to send an application message on the overlay, and it consequently has to know the current state of the network. If applications need to know if a node is currently running the service, it has to execute an explicit request to CrossROAD that will check the status of the selected destination through the cross-layer interaction.

In addition to the standard definition of the P2P CommonAPI, CrossROAD defines another class aimed at generating instances of CrossROAD objects (*Node*, *Endpoint*, *Id* and others) returning the correspondent interface. In this way, every applications programmer needs only to create an instance of the *InitCommonAPI* class to be able to implement interactions with the overlay. Through single functions implemented by this class, the application can directly act on CrossROAD objects only referring to methods of the correspondent interface.

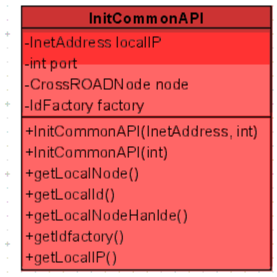


Figure 5.7: InitCommonAPI class diagram

Specifically, the object of type InitCommonAPI, created by the application at the startup, associates to the local IP address the service identifier and the port number related to the specific application. The constructor method can be called specifying the local IP address or selecting it from a configuration file. Since each instance of CrossROAD has to be associated to a specific application, we designed a configuration file, to be available in the current directory of CrossROAD, where it has to be specified the following information:

- IP address of the local node (dotted decimal string);
- Service description (string), service identifier (int), port number (int); for each service developed on top of CrossROAD a line of this type has to be defined. Each line is represented by those data separated by a blank space.

In addition the constructor creates an instance of CrossROAD Node to manage the overlay, and an instance of IdFactory to be able to calculate logical identifiers of messages' keys. Finally, through specific functions it recovers all information related to the CrossROAD node.
At this point the application is completely independent from the overlay management and data distribution policies; all this functionalities are directly implemented by the underlying node.
In particular, a CrossROAD node performs the following tasks:

- it establishes an interprocess communication with the XL-plugin in order to implement cross-layer interactions;
- when it is notified that at least another node takes part to the same overlay, it creates an instance of the Endpoint class, that directly calculates the overlay network routing table and generates two threads:
    - the RoutingServerManager, that waits for application messages. When it receives one of these messages it creates a child thread that processes its contents and consequently determines the best destination for the specified key value. In the

meanwhile, the main thread comes back waiting for application messages. In particular, before processing the message key value, the child thread has to check the consistency of the overlay data structures with the XL-plugin. Then, if the best destination, obtained by the current overlay routing table, is a remote node, the child thread establishes a TCP connection toward it and dies; instead if the local node results to be the best destination, it executes a call-back to the deliver function of the related application and consequently dies;

- the AliveServer, that communicates to the XL-plugin the number of the local port on which it can check if the instance of the local node is currently running.

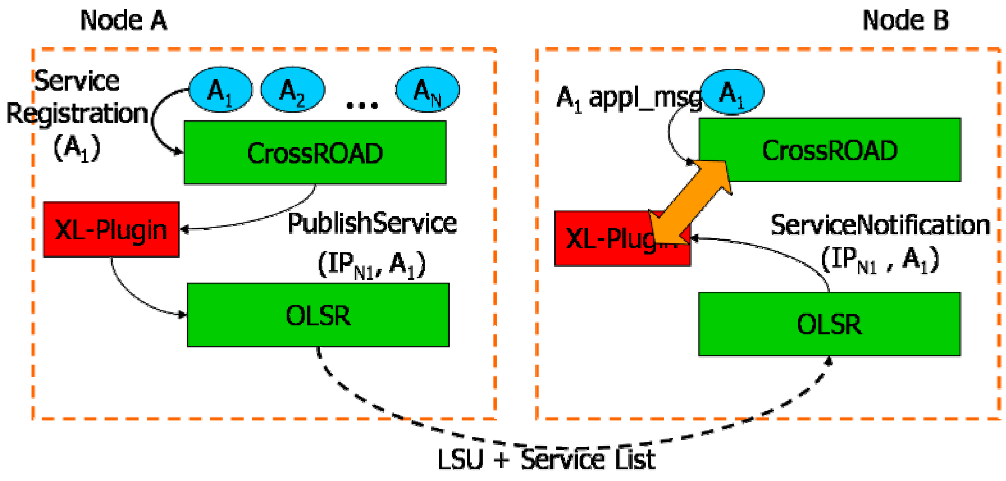


Figure 5.8: Interactions between CrossROAD, XL-plugin and OLSR

In order to better understand CrossROAD behavior and its optimization on ad hoc networks, a description of its interactions with the XL-plugin and the proactive routing protocol is shown in Figure 5.8. It represents a simple example of how cross-layer interactions can be exploited in the creation of the overlay between two nodes (A and B). The interaction between CrossROAD and XL-plugin starts when the application A_1, running on node A, registers the related service identifier creating a new instance of CrossROAD. In this way the local node joins the overlay sending to the plugin a message of *PublishService* containing its IP address and the service identifier associated to the specific application. When the plugin receives this message, it encapsulates that information in the first routing packet that will be sent on the network, and it stores this content in the *Local Services Table* (see Section 1.5), which maintains the list of services provided by the local node. On the other hand, when this message is received by another node (e.g. B), the plugin processes the additional information, and stores it in the *Global Services Table*, selecting all services currently provided by every node of the network. At this point, when the application decides to send a message on the overlay, first CrossROAD checks the consistency of its internal data structures with the plugin, and then it selects the optimal destination for that message between nodes currently running the overlay, and contact it through a simple p2p connection. In this way, all remote connections, needed by Pastry to build and maintain the overlay data structures, are eliminated, and every node of the overlay knows all the other participants, avoiding the multihop middleware routing introduced by the subject-based policy of Pastry. In order to implement this solution, a messages' exchange protocol has been defined between CrossROAD and the XL-plugin. Messages definition has been divided between messages from CrossROAD to XL-plugin and vice versa.

From CrossROAD to XL-plugin:

MessageType	Service Identifier	Port number
1 Byte	2 Byte	2 Byte

Where MessageType represents a code to identify requests of: *PublishService*, *LookupService*, *DisconnectService,* and *Alive*. The request of *PublishService* is sent by CrossROAD when the application starts running creating a new instance of the InitCommonAPI class. Consequently the plugin sends the message to the routing protocol to be added to the next packet and flooded on the network. Then, every time the application wants to send a message on the overlay (calling the Endpoint function "`route(Id id, Message message,NodeHandle hint)`"), CrossROAD sends a *LookupService* request in order to recover the current state of the overlay. In this case the message does not contain the port number since the plugin has only to read its internal data structure related to local services. Finally, when the application explicitly closes calling the Endpoint function "`closeApplication()`", CrossROAD sends a *DisconnectService* message to XL-plugin that will send it to the routing protocol. In addition, at the startup, CrossROAD has to send an *Alive* message to XL-plugin to notify the port number on which the specific instance of CrossROAD Node listens for plugin requests. Specifically, XL-plugin periodically tries to establish a local connection to CrossROAD to verify if it is currently working. If not, specifically in case of CrossROAD or application failures, the plugin automatically sends a *DisconnectionService* message to the routing protocol.

On the other hand, XL-plugin defines a message to be sent to CrossROAD that contains the list of nodes providing a specific service with the following format:

NodeList	Number of nodes	IP address	Port number	…	IP address	Port number
		(Node 1)			(Node n)	
1 Byte	4 Byte	6 Byte			6 Byte	

In addition, XL-plugin notifies an error message to CrossROAD in case another instance of CrossROAD related to the same service is already running on the local node. In fact multiple instances of CrossROAD for the same service are not allowed to maintain the consistency of plugin internal data structures (see Section 1.5).

5.7. Conclusions

CrossROAD represents an optimized middleware solution for ad hoc networks based on the cross-layer architecture. Even if it maintains basic principles of the p2p systems based on structured overlay networks, it enormously reduces the introduced overhead, thanks to the cooperation between the middleware and the network layers. Exploiting the entire knowledge of the network topology, CrossROAD completely eliminates the communication overhead related to the high number of remote connections necessary to the overlay management.
Currently CrossROAD software architecture is completely developed and a preliminary test phase has been set up in the CNR campus in Pisa. Specifically the same set of experiments developed for Pastry and analyzed in Deliverable D8 and in [AWSN], has been repeated running CrossROAD on top of Unik-OLSR v.0.4.8 enhanced with the XL-plugin. Experimental results pointed out that the network overhead has been enormously reduced by cross-layer interactions and the system is highly responsive to network partitioning and topology changes thanks to the use of a proactive routing protocol. Currently we are planning a new experimental phase during

next months in order to test and evaluate CrossROAD performances and behavior on ad hoc networks of medium-large scale, and also test distributed applications developed by other project partners to run on top of it.

References

[CGMT03] M. Conti, S. Giordano, G. Maselli, and G. Turi. Mobile-MAN: Mobile metropolitan ad hoc networks. In *Proceedings of the 8th IFIP Conference on Personal Wireless Communications (PWC'03)*, September 2003.

[CGMT04] M. Conti, S. Giordano, G. Maselli, and G. Turi. Cross layering in manets' design. *IEEE Computer, Special Issue on Ad Hoc Networks*, February 2004.

[CGT04] M. Conti, E. Gregori, G. Turi, Towards Scalable P2P Computing for Mobile Ad Hoc Networks, In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops,* March 2004

[DZDKS03] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for structured peer-to-peer overlays. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, February 2003.

[FreePastry] FreePastry web site. http://freepastry.rice.edu.

[KP02] R. Koodli and C. E. Perkins. Service discovery in on-demand ad hoc networks. Internet Draft, October 2002.

[MM02] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. *Proceedings of 1st International Workshop on P2P Systems (IPTPS '02)*, 2002.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[RFHKS01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM 2001*. ACM Press, 2001.

[SMKKB01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*. ACM Press, 2001.

[SRS01] C. A. Santivanez, R. Ramanathan, and I. Stavrakakis. Making link-state routing scale for ad hoc networks. In *Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'01)*, 2001.

[ZKJ01] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. *Technical Report UCB/CSD-01-1141*, UC Berkeley, April 2001.

[Del05] F. Delmastro, "From Pastry to CrossROAD: Cross-layer Ring Overlay for AD hoc networks", In *Proceedings of Mobile Peer-to-Peer 2005 Workshop, in* *conjuction with PerCom 2005 Conference*, Kauai Island, Hawaii, Mar. 2005

[Uni] A. Tonnessen, OLSR: Optimized link state routing protocol. Institute for Informatics at the University of Oslo (Norway). Available: http://www.olsr.org

[AWSN] E. Borgia, M. Conti, F. Delmastro, and L. Pelusi, “Lessons from an Ad hoc Network Test-bed: Middleware and Routing issues”, Ad Hoc & Sensor Wireless Networks An International Journal, Vol. 1 N. 1-2, 2005.

6. APPLICATIONS

6.1. Content sharing: UDDI Approach

Main characteristics of Mobile Ad Hoc Networks (MANETs) are the lack of infrastructure and a topology highly dynamic. However, it is clear that, in the near future, as technologies for mobile ad hoc networking will become available and stable, the users will require to access services in mobile ad hoc networks as it is currently possible on fixed networks and WLANs. For that reason, service discovery protocols and delivery mechanisms will play a strategic role for the exploitation of ad hoc networks. However, there may be problems with how resources can be shared in a dynamic, ad-hoc environment, where both servers and clients are ephemeral. Existing service discovery protocols and delivery mechanisms, that Netikos, during first months of activity, has investigated, designed for fixed networks with centralized service management, fall short of accommodating this complexities of the ad-hoc environment. They also place emphasis on device capabilities as services rather than device independent software services. So there are several efforts toward new service discovery and delivery protocols designed specifically for ad-hoc, peer-to-peer networks, and targeted for device independent services.
In our activity, we propose a service discovery and location protocol for MANETs called *UDDI for manets (UDDI4m).* This protocol exploits the traditional UDDI protocol with the introduction of an interfacing level that allows fitting into ad hoc environment, considering the features of this type of networks. Even if the starting point is the existing UDDI protocol [UD02], our service discovery protocol distinguishes from other works, including: WSDL /UDDI approach with web (and grid) services [IF03] in the IETF WGs, such as zero-conf [GN03] and Service Location Protocol (SLP) [G99], and other approaches such as plug-and-play hardware discovery [KF02].
In the literature, there are also many discovery service protocols for the wireless networks, but these protocols are mainly based on the presence of infrastructure as Access Points, Home/Foreign Agent in the case of MobileIP. In MANETs each nodes is at the same level, so it is critical to have a node that plays the role of the centralized server, for several reasons. Each node is dynamic and volatile, nodes can be power constrained, and there is no central authority that guarantees on the "goodness" of a node. Thus, the lack of infrastructure requires a dynamic service discovery protocol.
In this view, each node must be:

- client, to request the use of a service;
- server, to provide the requested service;
- router, to forward the requested service between two nodes.

In order to obtain that using the traditional UDDI protocol, the UDDI4m introduces a layer between transport and application levels. This layer provides a service location distributed among all the nodes in the network that are active at a given moment. In particular, the objective is to obtain an UDDI service similarly to fixed networks, but distributed among the nodes of the network.
In classic UDDI there is a central server that manages the UBRs (UDDI Business Registers) distributed among network's nodes. So, a client request is accepted and managed from the central server that retrieves the list of available services requested from user in its database and reply it to the user. In case of ad hoc network, it is not feasible to overcharge a single node with the management of the global database. The additional level of UDDI4m is composed by an overlay network of local servers, each of them running on the client device. Each client request is handled from its local server that becomes the server that is in charge of discovering the available network's services. This discovery is implemented through the communication with other local servers currently running the overlay network. The communication between servers is represented

by a peer-to-peer connection, as opposite to the legacy UDDI paradigm, which is based on the client-server mechanism.
The proposed architecture has several advantages:

- We perform service discovery and location using standard web service model; thus it suitable to both ad hoc and legacy networks. Furthermore, it is fully compatible with existing service discovery and location protocols.
- The web service model is enriched by the presence of a structured overlay network, adequate to MANET characteristics.
- We perform service discovery and location not simply related to full services, but also for accessing their details; thus implementing also services that can be difficultly offered in MANETs.

6.1.1. UDDI Overview

The web services community has defined the Universal Description, Discovery and Integration (UDDI) specifications and its associated protocols to provide a database of services available [UD02]. UDDI defines a way to publish and discover information about web services. The term "Web Service" describes specific business functionality exposed by a company, usually through an Internet connection, for the purpose of providing a way for another company or software program to use the service.
UDDI relies upon a distributed registry of businesses and their service descriptions implemented in a common Extensible Markup Language (XML) format. The core component of the UDDI project is the UDDI business registration, an XML file used to describe a business entity and its Web services.
The information provided in a UDDI business registration consists of three components:
white pages, including address, contact and known identifiers;
yellow pages, including industrial categorization;
green pages, technical information about services exposed by the business. This page includes references to specifications for Web services, and it supports pointers to different URL based discovery mechanisms if required.
The UDDI Business Registry (UBR) can be used at a business level to check whether a given partner has particular Web service interfaces, to find companies in a given industry with a given type of service, and to locate information about how a partner or intended partner has exposed a Web service in order to learn the technical details required to interact with that service.
XML simplifies the exchange of data, solving integration and interoperability problems. XML provides a cross-platform approach to data encoding and formatting. SOAP (Simple Object Access Protocol), which is built on XML, defines a simple way to package information for exchange across system boundaries. SOAP bindings for HTTP are built on this packaging protocol and define a way to make remote procedure calls between systems in a manner that is independent of the programming language or operating system choices.
The UDDI specifications consist of an XML schema for SOAP messages, and a description of the UDDI API specification. Together, these form a base information model and interaction framework that provides the ability to publish information about a broad array of Web services.
The core information model used by the UDDI registry is defined in a XML schema. XML was chosen because it offers a platform-independent data model and it allows describing hierarchical relationships in a natural language. The UDDI XML schema defines 4 types of information that represent possible typologies of data that a technical user would need to know in order to use a partners Web service (see Figure 6.1).

The **Business information** is related to the business unity, technical staff, programmers or application programs. To support publishing and discovering operations about a business, the UDDI Business Registry maintains these information in a structure named *"businessEntity"*. The information includes support for "yellow pages" taxonomies. The **Service information** is technical and business descriptions of Web services (e.g. the "green pages"). They are contained in two data structures: *"businessService"* and *"bindingTemplate"*. The *businessService* structure is used to group Web services referring to types of services. Technical Web service descriptions are contained in the *bindingTemplate* structure. It contains information relevant to applications that need to connect to a remote Web service. The single element of the *bindingTemplate* structure contains a list of references to Web service's specifications. These references form a technical fingerprint that can be used to recognize a Web service that implements a particular behavior or a programming interface. An UDDI registration for a service consists of the definition of an entry for the business partner data structure, a logical service entry that describes the purchasing service, and a *bindingTemplate* entry that describes the purchase order service by listing its URL and reference to a "*tModel*".

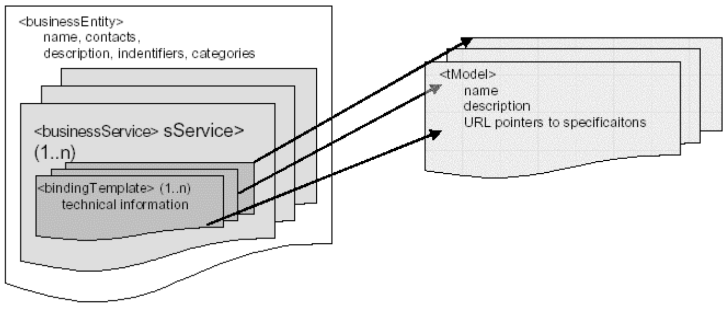


Fig. 6.1 data structure used in the UDDI standard to describe and discover information about Web services

TModel information is a *metadata* defined to describe a service, including its name, publishing organization and URL pointers to the actual specification.
The UDDI specifications include also the definition for Web service interfaces that allow programmatic access to the UDDI registry information.
The API provided from UDDI standard protocol is divided into two logical parts: "Inquiry API" and "Publish API". The Inquiry API consists of two parts: one used to build programs providing the research and browsing of information collected in a UDDI registry, and the other used in case of Web service failures. Programmers use the Publisher API to develop interfaces for the interaction with an UDDI registry.
The Simple Object Access Protocol (SOAP) is a way to use XML and HTTP to create an information delivery and remote procedure mechanisms. Several companies submitted this draft note to standard RPC conversation on the World Wide Web. The draft note describes a specification that is useful to describe a Web service.
The general scenario for using UDDI is considering the preparation required to write a program that uses a specific Web service, the step to follow are:

- the programmer, which writes a program that uses a remote Web service, uses UDDI business registry to locate the *businessEntity* information.
- the programmer either drills down for more detail about a *businessService* or requests a full *businessEntity* structure.
- the programmer prepares the program based on the knowledge of the specifications for the Web service. This information may be obtained by using the *tModel* key information contained in the *bindingTemplate* for a service.
- at runtime, the program invokes the Web service as planned using the *bindingTemplate* information.

6.1.2. UDDI in Ad Hoc Network: UDDI4m

The *UDDI for manets* (UDDI4m) is a protocol for service discovery and location for MANETs. This protocol exploits the traditional UDDI protocol optimized for an ad hoc environment.
In MANETs, it is critical to consider a single node that plays the role of centralized server due to the dynamic nature of that environment. In fact each node is dynamic and volatile, nodes can be power constrained, and there is no central authority that guarantees on nodes behavior and there is no knowledge of services provided by every single node. Thus, the lack of infrastructure requires a dynamic service discovery protocol.
The UDDI4m can be considered as a protocol and it is located in an intermediate layer between middleware and application. Based on an overlay network that guarantees a fair balancing of workload and data distribution on network nodes, UDDI4m provides a service location distributed among all active nodes.
Considering a peer to peer network, each node has the same role and cooperation's policies are defined to guarantee the service. Considering a dynamic service discovery protocol as UDDI, each node not only can publish its own services, but it can also assume the role of server, maintaining, in a local database, information related to services provided by other nodes. In ad hoc networks, characterized by heterogeneous nodes, not all of them can maintain a local database, but at the same time they can locally maintain information related to their own services. For this reason we consider two different classifications of nodes in the network.
First classification is relative to node that can provide or use a web services:

- **provider nodes:** are nodes that provide a service, they are able to publish contents relative to the service also on other nodes of the network.
- **user nodes:** are nodes that want to use a web service. They use the UDDI4m service to retrieve the information needed to invoke the related web service.

In Figure 6.2a is showed this classification where the nodes can be: only *users* of a service (**U**), only *provider* of a service (**P**) or both *user* and *provider* (**P/U**).
The second classification is related to UDDI4m service. Each node must have the client and server side of the UDDI4m service. The server side manages the publishing and inquiry requests from the client side, so it recovers the contents (according to the client request) from the server-side of the nodes participating to UDDI4m service:

- **nodes with UBR4m:** are nodes with adequate resources to manage an UBR4m registry for the UDDI4m service. These nodes are indicated with **C/S**.
- **nodes without UBR4m:** are nodes that can't manage an UBR4m registry due to limited resources. They are indicated with **C**.

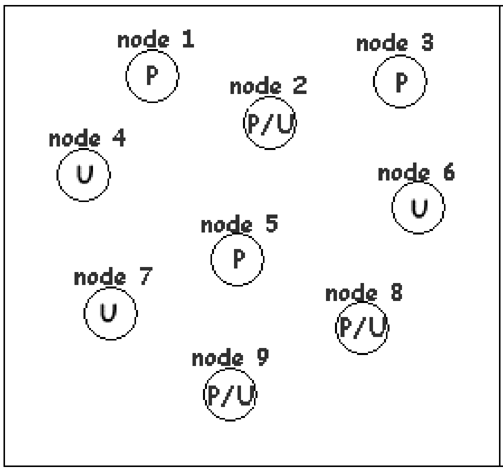


Fig 6.2a

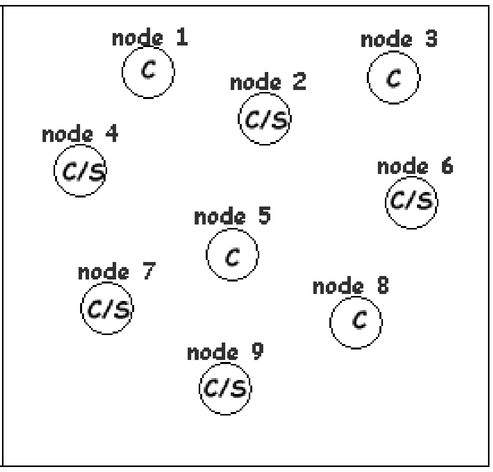


Fig 6.2b

Two classification type of the nodes presented in ad hoc network

The second classification is showed in Figure 6.2b by considering the same network topology of Figure 6.2a. In Figure 6.2b a node must be client to participate to UDDI4m service, but may or may not have the UBR4m registry. So, we call *server UDDI4m node* a node with UBR4m, and indicate it with **C/S**; nodes without UBR4m are named *client UDDI4m node* and are indicated with **C**.

The two classifications are independent, for example, a **provider node** of a service can be **a node with or without UBR4m**; similarly a **user node** (see Figures 6.2a, 6.2b).

Each node do not care in which node are stored the information related to its own service; it can be stored on the node itself, or on another one.

Each client request is handled from its local server part, so the local server becomes the server that is in charge to discover the available network's services: the local server forwards the client request through the overlay network. This discovery is realized communicating with other servers taking part to the overlay network. This communication is represented by a peer-to-peer connection with the node that probably maintains the selected service, using a proximity logic based on the service identifier.

In mobile ad hoc environments, each node cannot store all available services in its own register due to several reasons:

- **Topology changes:** the contents become obsolete in short time due the high mobility and frequent changes of the network's topology. So, the updates and accesses on registers must be quick and practical.
- **Energy constraints:** in ad hoc network there are devices with limited resources (e.g. PDAs,). Thus, the register cannot contain many data. Furthermore, on some devices, there are not enough resources to run the service discovery and location protocol.
- **Communication constraints:** in ad hoc networks communications are costly and there is not certainty of success, thus communications for maintaining updated contents must be minimized.
- **Security**: in ad hoc networks there is no certification authority for guaranteeing that a node is trustable. Thus is not feasible to centralize service discovery and location to an unknown node.

Thus, in Mobile Ad Hoc environment, a service discovery protocol must be decentralized and distributed in the network, it must minimize the information maintained at each node and the communications among nodes, and must deal with the network topology changes. In UDDI4m, this is performed by enforcing the classic UDDI by the introduction of an overlay network, which

provides the communication structure for distributing the service discovery and allows dealing with the nodes' mobility.
The UDDI4m differs from the UDDI standard due to the characteristics of the ad hoc environment; in the latter not all the features can be implemented.
In the UDDI4m the information type provided is the *green pages*, the technical information about service that are exposed by the business, that in this case is a node taking part to the Ad Hoc Network.
The UDDI Business Registry for Manet (UBR4m) is lighter than the standard UBR, and the UDDI4m service provides only publication and search procedures.
In the UDDI standard, the IP addresses of UDDI servers are known a priori. In the ad hoc environment, this knowledge is missing because there isn't a centralized server that manages the clients' requests; in this case the server is decentralized and distributed on the peer nodes of the network that support the UDDI4m service.
The idea is to use a middleware layer based on structured overlay that manages the content distributed in the network related to an UDDI4m service. In order to distribute contents (through subject-based routing policies) related to the service provided from peers, we must define a categorization of the services, for example: File Sharing, Chat, Device, E-commerce, etc.
The nodes that form the overlay network must have policies to regulate the cooperation among them. Cooperation among nodes is an important characteristic of both peer-to-peer and MANETs. The UDDI4m provides features to implement several policies of cooperation management. These policies are realized by the service Access Control List (ACL) based on the user behavior or profile. However, if the middleware layer is not present, UDDI4m service must recover the information stored on the nodes that form UDDI4m service by message broadcasting to obtain the IP addresses and then contact each single node, thus significantly increasing the network overhead.
Moreover, the *UDDI4m service* classifies the services available in the network, and allows accessing the contents presented in the related service descriptions. Thus, through the *UDDI4m* service it is possible to design a search engine in the Ad Hoc network.
The proposed solution, based on the use of a structured overlay optimizes features of the UDDI4m service in ad hoc networks. Specifically, the *UDDI4m service* can be used on a legacy architecture running the standard Pastry protocol [FP], or it can be used on top of a cross-layer architecture, where a new middleware solution has been designed exploiting cross-layer interactions with a proactive routing protocol providing the uniform distribution of data between peers (CrossROAD [D05]). For more details on CrossROAD functionalities see Chapter 5. In Figure 6.3 the introduction of a Service layer, represented by the UDDI4m service, in legacy and cross-layer architectures is shown. The behavior of UDDI4m service does not change in these two different architectures, but it is aware of the presence of which overlay network is currently in use, and its performances considerably change.

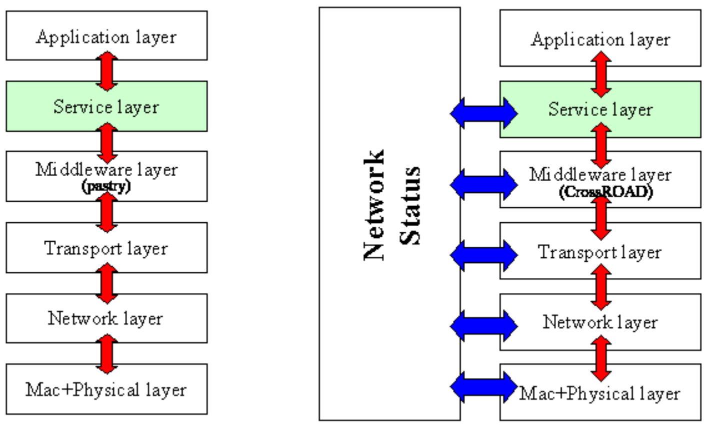


Figure 6.3 on the left is showed the interaction of the service layer in the legacy architecture, on the right in the MobileMAN architecture

6.1.3. UDDI4m on CrossROAD

In this paragraph we describe and figure out features and advantages of the interaction between UDDI4m service and a new middleware solution for ad hoc network (CrossROAD).
As already explained in previous sections, CrossROAD exploits the cross-layer interaction with a proactive routing protocol to maintain a complete knowledge of the network topology and to autonomously establish and manage all data structures of the overlay network. Specifically, the UDDI4m service interacts with CrossROAD implementing part of the p2pCommonAPI as a simple distributed application. In this way, each node running UDDI4m takes part to the overlay network following its routing policies, and it can use it to efficiently locate the node that maintains specific information even if the network topology is continuously changing. In fact one of the main hypotheses of the standard UDDI, developed for the wired network, was represented by the knowledge a priori of a set of servers where publishing and recovering services information. In ad hoc networks, where the network topology is highly dynamic and nodes interactions are temporary, it is not possible to have a list of server nodes and to be sure of their availability. For this reason, the presence of an overlay network where each node has at least a partial knowledge of nodes taking part to the same service, and a policy to establish a possible destination for each request is defined, can optimize UDDI4m performances. In addition, as described in Section 6.1.2, UDDI4m nodes are characterized by different functionalities: **UDDI4m clients** can publish and recover services information, but they don't have a registry to store contents; **UDDI4m servers** handle a registry where information related to published services are stored, in addition they can also assume the role of client to publish and require other services information. In the overlay network all nodes have the same characteristics, physical constraints are not specified and they are uniquely identified by a logical address that can represent the optimal destination for a data to be distributed through the subject-based routing. In case of UDDI4m, client nodes are not able to receive a publish request from another node because they don't have the registry where to store it. For this reason, all requests that involve data storage cannot be delivered to a client node. To solve this problem, also at the middleware layer a node categorization is needed. Since CrossROAD overlay network is represented by a

160-bit circular address space, it is possible to divide this logical space in two parts applying a mask to logical identifiers in order to distinguish server nodes from client nodes. This represents an enhancement to standard features of overlay network that can enormously optimize UDDI performances on ad hoc networks.

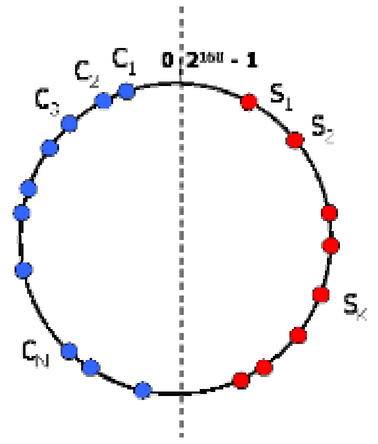


Figure 6.4 CrossROAD overlay designed for UDDI4m

In Figure 6.4 an example of overlay network consisting of several client and server nodes is shown. In order to obtain a configuration of this type, it is necessary that UDDI4m application running on a particular node, specifies if that node assumes the role of client or server. In this way, when the CrossROAD instance is created, it knows how to compute the local node identifier and which types of requests can be accepted from that node. Specifically, CrossROAD implements the subject-based routing on this particular overlay, applying the same mask of the server nodes also on key values used to distribute data on the network. In this way all requests of storage or recovery of services information are forwarded only to one of the server nodes, that one with the logical identifier numerically closest to the key value after the mask has been applied on it. For this reason, also the definition of key values for data distribution is an important feature of UDDI4m. Since UDDI4m requests are mainly represented by publication and recovery of services information, the definition of services categories is necessary to exploit the subject-based policy of a structured overlay network. Hence, we define the key value as the type of service to be stored in the UBR4m registry or to be retrieved. In this way, each server is in charge of maintaining information related to a specific service category, that one identified by the logical identifier numerically closest to its own. In addition, in order to guarantee the system reliability in case of nodes failures, a policy to store replica information on other server nodes has to be defined, and it is currently a work in progress.
We can suppose to define a set of macro-services as Content Sharing, Chat, Video/Conference, Devices, E-Commerce, Mail Server, Forecast Information, and others, foreseeing additional specialization in sub-services (e.g. a Content Sharing service can be divided in mp3, movies, games, documentation, and others). In this way, UDDI4m has not to know the server list a priori, but it has only to implement the p2pCommonAPI in order to exploit CrossROAD advantages. When a client node decides to publish a service, it specifies the category of the service as key value for the CrossROAD message, and the same for a request of update or recovery of services information. Then CrossROAD selects the best destination for each message checking every time the consistency of its internal data structures with the current state of the network topology. Each message is sent to a destination currently available on the network, avoiding all attempts to connect to all server nodes looking for the one maintaining the required information, as in the standard implementation of UDDI. In next sections UDDI4m software architecture will be

presented referring to the presence of an underlying overlay network that can be CrossROAD, in case of cross-layer architecture, or Pastry using legacy architecture.

6.1.4. UDDI4m Software Architecture

As described in the previous section, we introduce a new layer between middleware layer and application layer that we call *Service Layer*. This layer performs the communication among the application and the overlay network, allowing retrieving the information about the UDDI4m service exploiting the middleware capabilities, and thus performing the discovery and location service in the ad hoc network as an information retrieve over the overlay peer-to-peer network. The legacy stack augmented with the new service level interacts with middleware layer to have an overlay network of the service and with the application layer to export functionality to discover and publish services.

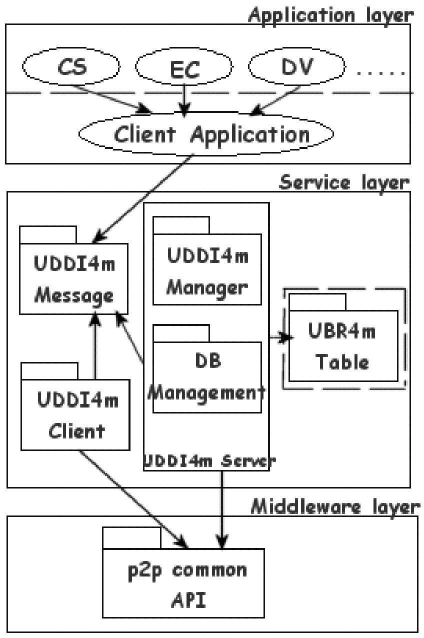


Figure 6.5 Interaction among the Application, Service and Middleware layers

The interaction among the three layers involved in the service discovery (*application layer*, *service layer*, *middleware layer)* is explained in Figure 6.5. The *Service layer* exports the API offered by the UDDI standard specification over the overlay ring (UDDI4m API), these API are implemented from *UDDI4m module*. Also, the service layer registry used to store the contents (*UBR4m*) is compliant with UBR registry of UDDI standard but some simplifications are adopted to have a light platform considering the features of application environment.
The UDDI4m nodes form an overlay network of the service; this ring is obtained using a middleware protocol, so the advantages inherited from the lookup protocol are exploited from the *UDDI4m service*. For example the number of exchanged messages among themselves has a low complexity. The nodes that have only the client side participate to the overlay network of the UDDI4m service to publish and retrieve information.
This architecture is fully modular and the interaction between the service layer and middleware layer can be abstracted and made independent from middleware by introducing an interface module between UDDI4m service and the middleware. This interface directly interacts with the

various *p2p common API* implemented by Pastry or CrossROAD package. Hence, the *UDDI4m service* can be used both on legacy architecture and on cross-layer architectures. The *UDDI4m service* loads the module according to the corresponding middleware.
The software architecture of the UDDI4m service is composed by the following modules:

- The *UDDI4m client* generates requests to the *UDDI4m server* module using the ID-Number; it connects to the (local or remote) *UDDI4m server* to publish or recover contents. The server node is selected in according to the services' categorization service. The communication among peer nodes is represented by a p2p connection.
- The *UDDI4m server* side is divided in two parts: one block (*UDDI4m Manager*) that implements and provides the API to publish the information on databases (*publishing API*) and to retrieve the information from databases (*inquiry API*), these API are used from *UDDI4m_Service* module that is the core of the UDDI4m service because it implements the methods of the service: *publishService*, *findService*, *updateService* and *deleteService*; a block (*DB Management*) that implements the data structure.
- *UBR4m Table* module directly manages the database translating the client request in queries on database and recovering the requested data. This module is optional, so it cannot be installed on all nodes of the network, e.g. devices with resource constraints, like IPAQs, cannot manage the *UBR4m Table*, therefore if these devices want to publish their services they must publish them on other nodes.
- The *UDDI4m Message* module implements the messages exchanged among nodes in the network through the middleware layer.

The messages between service and middleware layer are divided into four types. They have three fixed fields (see fig. 6.6): *typeService* that is the key of categorization used to exploit the functionalities of the structured overlay; *typeMessage* is the type of message that can be: *publish*, *find*, *delete* and *update*; *number* is the number of replicas, this field is used to manage the backup copy of the content relative to service.

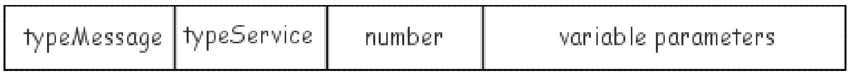


Figure 6.6 structure of the messages exchanged between UDDI4m nodes using overlay network

Specifically, for each type of message there are variable parameters:

- *publish message* is the message sent from *client UDDI4m node* in the network when it wants to publish the information relative to its own service. The variable parameters are *BusinessService*, *BusinessEntity*, *Contact*, *BindingTemplate* and *TModelInstanceInfo*, corresponding to all data structures to be stored in the UBR4m registry;
- *find message* is the message sent from *client UDDI4m node* in the network when it wants to recover information related to services' categorization or to a specified service. The variable parameter of this message are: *serviceKey* the key of the service for which the node wants recover the information; if this key is null, then information related to services' categorization are retrieved; *type* is the categorization type of the service; *IpClient* is the IP address of the client that has sent the request. As a reply, the *server UDDI4m node* sends the list of nodes that provide the selected service or service categorization. This reply is directly sent to the requesting node exploiting the (known) client IP address.
- *delete message* is the message sent from node when it wants to delete a service that previously it has published. The variable parameter is the key of the service to delete (*serviceKey*)
- *update message* is sent from a node when it wants to modify the information relative to a service it provides. The variable fields are related to the information to update: *service key* is the key of the service to modify, *BusinessService*, *BindingTemplate*, *TModelInstanceInfo*.

Each device includes a module at the application layer that simplifies the human task when publishing and inquiring services in the ad hoc network, called *Client Application*.
This Client application offers to the user a user friendly GUI (Graphic User Interface): an html page to manage the *UDDI4m service*. To invocate the web services available in the network, the system uses the SOAP protocol based on XML language, in order to be compliant with the standard UDDI specifications. This latter does not use the java-based API for accessing an UDDI registry, but define a series of SAOP messages that UDDI registry can accept. The *UDDI4m API* has two roles:

1. It completely hides this complexity to the application layer, and allows interacting with UDDI4m registry without knowing the SOAP or the XML messages and the data structures that UDDI interacts with.
2. It communicates with the middleware below (through the *p2p common API)* to obtain information on the peer nodes that support the *UDDI4m service* and sends messages to retrieve the contents stored in the databases.

Generally, using a p2p system like CrossROAD or Pastry, if the request represents a lookup for a specific service, the requiring node forwards the request to the node taking part to the overlay that has the logical address closest to the logical identifier of the research key.
Particularly, using CrossROAD that exploits main features of cross-layer architecture, the requiring node directly establishes a remote connection with the node that maintains the service's information exploiting the proximity logic between logical addresses and research key. In this

case the complexity of the lookup procedure implemented by the overlay is drastically reduced to a constant cost, because every node of the overlay directly knows the other participants and it can autonomously calculate the best match between logical address and research key.
In this way also the Service Discovery protocol is optimized to run in an ad hoc environment.

6.1.5. Implementation

The *UDDI4m service* is implemented with a modular and light approach. We simplified the data structure, their relationships, and the protocol to manage them (API) since the complexity of standard UDDI is not necessary and not appropriate to the way services are provided in the ad hoc networking environment, as detailed below. This simplification resulted in minimizing the memory occupancy and in reducing the communication, essential requirements for service discovery in ephemeral and low power nodes.

Data Model
Considering the focus of this service discovery we implemented a *lighter* version of the UDDI standard. The subset of the UDDI database data structure that are used to categorize the information in UDDI4m are:

- *Business Entity* that describes the entity (business provider) providing the services;
- *Business Service* that describes the service provided from a *Business Entity*;
- *Binding Template* is a technical description of the service, in particular it contains the url where the service is available;
- *TModel* and *TModel_Descr* are template services. Those data structure provide a structure that allows re-use and, thus standardization within a software framework

Each *Business Service* has a link with *TModel*, when a new *Business Service* is inserted and the relative *TModel* if not present is created.
We further simplified the data relationships and the data management:

- the *Publisher Assertion* is deleted, we do not consider the relations between two parties (provider nodes), as the network is heterogeneous and it is very rare that two nodes are in such a relationship;
- each Business Service corresponds to an only Binding Template;
- the UUID (Universal Unique IDentifier) technique to generate univocal key is substituted with a simpler algorithm that generates the keys with the information: (*date, local time and name of the entity*);
- the authorization level is omitted because, usually, the client and server are on the same machine;
- the contents of the data structures are simplified (figure 4);
- detail changes of the data structures: the *IdentifierBag* and *CategoryBag* fields are deleted because this information, in the UDDI standard, are used for speed searches of the web services and so in our application do not bring more advantages. The *lang* field relative to the adopted language is deleted because is supposed that the used language is only English language. This choice is reasonable because the environment is local and is not wide as the internet web world.

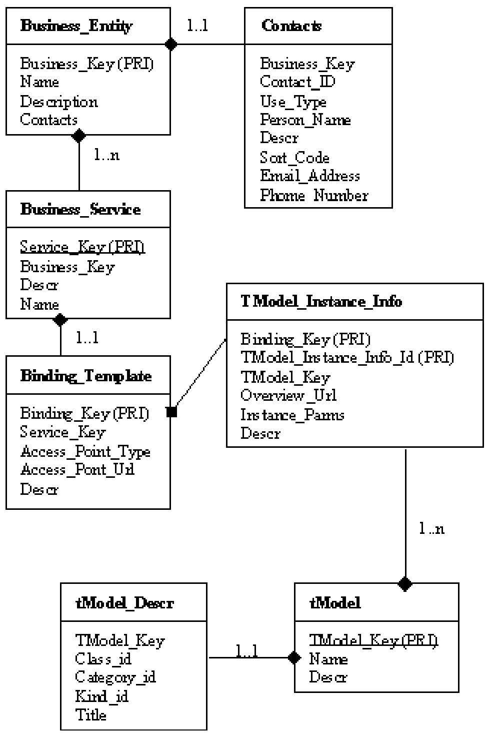


Figure 6.7 Relational model UDDI4m database of MobileMAN

The relational model of the MobileMAN database is showed in the schema in figure 6.7: in such model one Business Entity contains one or more Business Service entities; one Business Service contains one instance of the Binding template entity, which contains information relative to the instance of the TModel structure.
The relationship between *Business Entity* and *Business Service* is 1 to n because a *Business Entity* can have one or more service to publish.
The relationship between a service and its *TModel* template is implemented through the *TModel_Instance_Info* entity.

The *client application* shows to the user the html pages to give (limited) knowledge about services as: user-friendly name of the service, service type, symbolic name of the device offering the service. Based on this information, the user may seek more information about service: following the *Access_Poin_Url* field of the *Binding Template* entity he obtains more details of the service.

publishing/inquiry API
The publishing and inquiry API are compliant with the UDDI standard.
The publishing API are invocated to insert, delete or update information contained in the UDDI4m databases, so those API manage and modify the contents of the UDDI4m databases and are:

- *Save_Business()* to insert a new *Business Entity* if this entity does not exist or to update the existing *Business Entity*;
- *Save_Service()* to insert a new *Service Entity* if this entity does not exist or to update the existing *Service Entity*;

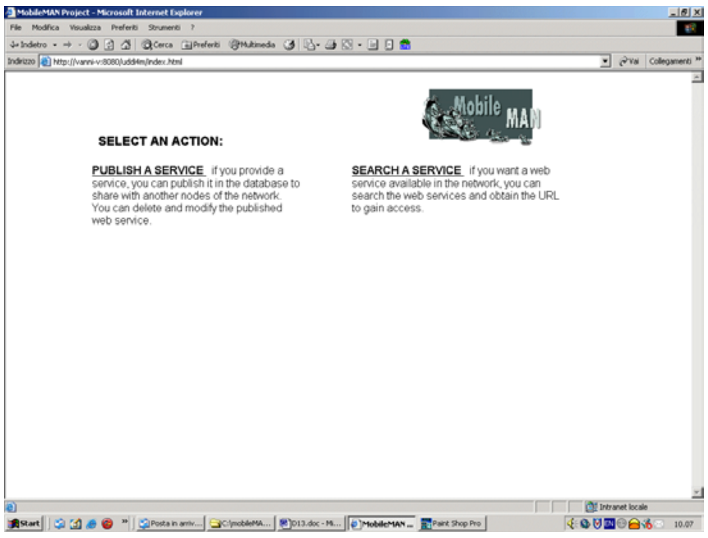
- *Save_Binding()* to insert a new *Binding Template Entity* if this entity does not exist or to update the existing *Binding Template Entity*;
- *Save_TModel()*to insert a new *TModel Entity* if this entity does not exist or to update the existing *TModel Entity*;
- *Delete_Business()* to delete a Business Entity;
- *Delete_Service()* to delete a Service Entity;
- *Delete_Binding()* to delete a Binding Template Entity;
- *Delete_TModel()* to delete a TModel Entity.

The inquiry API does not manage the registry, but make only the inquiries on the tables of the UDDI4m databases. They are:

- *Find_Business()* searches the *Business Entity* with BusinessKey value;
- *Find_Service()* search the services with the ServiceKey value or with the BusinessKey value listing all the services provided by that *Business Entity*;
- *Find_Binding()* searches the *Binding Template Entity* with the BindingKey or ServiceKey;
- *Find_TModel()* searches the *TModel Entity* with TmodelKey.

Prototype

The implementation has been done in JAVA, and runs on a laptop with Linux Operating System. The nodes, that have the server side, use Tomcat Web Server (Jakarta-tomcat-4.1.24). The database schema used is MySQL (server version 4.0.18-standard). The SOAP protocol is supported by the Axis package version 1.2.
In this implementation, the overlay network used is CrossROAD.
When the service is started for the first time, the database and the related tables are created. The *client application* shows to the user the html home page, where she/he can choose the action: service publishing or service retrieving (see Figure 6.8)


MobileMAN Project - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti Multimedia
Indirizzo http://vanni-v:8080/uddi4m/index.html
Vai Collegamenti
Mobile MAN

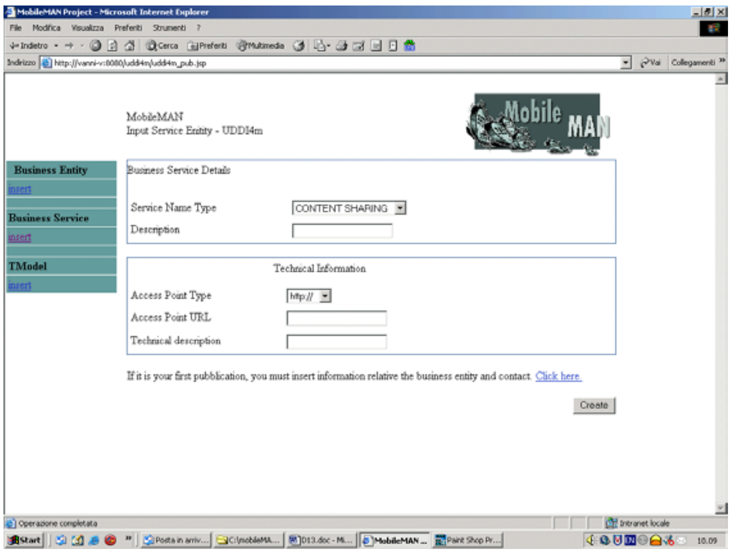
SELECT AN ACTION:

PUBLISH A SERVICE if you provide a service, you can publish it in the database to share with another nodes of the network. You can delete and modify the published web service.

SEARCH A SERVICE if you want a web service available in the network, you can search the web services and obtain the URL to gain access.

Figure 6.8 home page showed when the user consumer the UDDI4m service

In according to the selected choice, the *client application* proceeds to a different html page that shows different options to the user. For example, if the user wants to save a service, the *client application* shows the html page (see Figure 6.9), where she/he can insert the information related to the *Service Entity* and *Binding Template*. Once all the information has been inserted, the user can click on the 'create' button and a *publish message* containing the information relative to service to be stored is sent in the network exploiting the overlay. The node receiving the message stores the contents in the corresponding tables of its own UBR4m registry (Business_Service, Binding_Template, Contact and TModel_Instance_Info tables).


MobileMAN Project - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti Multimedia
Indirizzo http://vanni-v:8080/uddi4m/uddi4m_pub.jsp
Vai Collegamenti
Mobile MAN
Operazione completata
Intranet locale
Start Posta in arriv... C:\mobileMA... D13.doc - Mi... MobileMAN ... Paint Shop Pr... 10.09

MobileMAN
Input Service Entity - UDDI4m

Business Entity
insert
Business Service
insert
TModel
insert

Business Service Details

Service Name Type	CONTENT SHARING
Description	

Technical Information

Access Point Type	http://
Access Point URL	
Technical description	

If it is your first pubblication, you must insert information relative the business entity and contact. Click here.

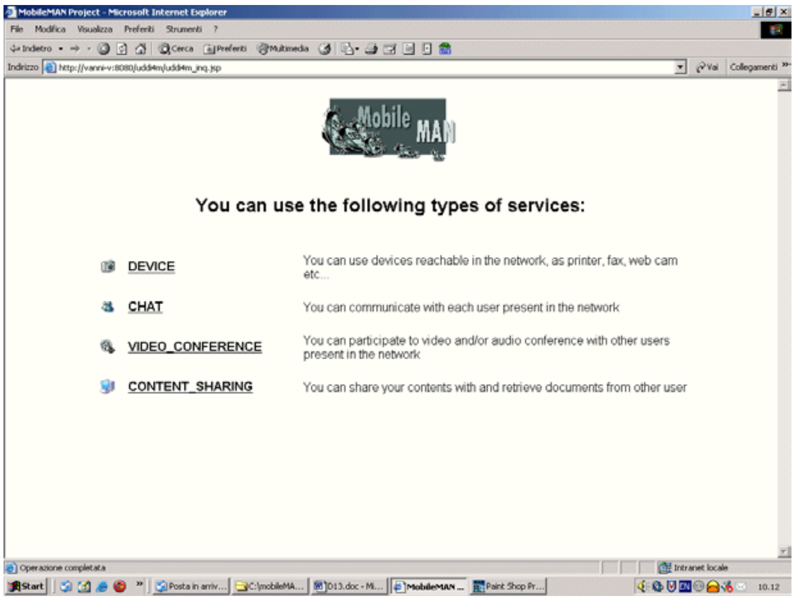
Create

Figure 6.9 html page shows a form where a user can insert the contents of the service to store

When the user wants to find a service, the *client application* proceeds to another html page (see Figure 6.10), where it is showed the list of the services available in the network, the user chooses the service that wants to use. When she/he clicks on the type of service the *client application* send a *find message* on the overlay

For example, if we choose a macro-service, i.e. Content Sharing, the search key is **CS**, and we suppose that a device wants to retrieve the URL of the nodes that provide this service with **CS** key. So the device invokes the *send(find_service(), CS)*, this message is delivered to the UDDI4m server node that has the *NodeId* closer to CS key. The destination node replies to the sender with a message containing the list of nodes that provide the content sharing service. The reply message is directly sent to the requesting node without using the middleware. In this way, the application knows the information (i. e. URL) of the UDDI4m server nodes that provide the Content Sharing service, so it's possible to invoke the methods of that service. These methods are invoked by SOAP protocol through http requests without using the CrossROAD protocol.

As a first approach, we assume that all nodes in the network are UDDI4m server nodes (all nodes have the UBR4m registry).


MobileMAN Project - Microsoft Internet Explorer

Indirizzo http://vanni-v:8080/uddi4m/uddi4m_inq.jsp

You can use the following types of services:

DEVICE	You can use devices reachable in the network, as printer, fax, web cam etc…
CHAT	You can communicate with each user present in the network
VIDEO_CONFERENCE	You can participate to video and/or audio conference with other users present in the network
CONTENT_SHARING	You can share your contents with and retrieve documents from other user

Figure 6.10 html page shows the list of service available in the network

6.1.6. Conclusion

The use of UDDI as service discovery and information retrieval system enriches MANET architecture with features that are typical of infrastructure-based network services.
The approach we propose supports the peer-to-peer cooperation; it is easy to implement and to be used. The user friendly GUI allows to easily publish owned information for sharing them in the network. The UDDI4m architecture makes this information quickly available to other nodes, fitting the main requirements of dynamic networks like MANETs.
Our approach allows sharing information via publishing on a local server, or on a neighbour one, using the middleware layer, if any, that provides features to manage the topology information and to keep the neighbour nodes ring up to date.
This approach has strong advantages for MANETs, as allows the use of standard objects that can be used on legacy architecture too, and does not require dedicated components for managing the topology information at application layer.
The work in progress includes the integration of our approach in the cross layering architecture of MobileMAN project [CGMT04]. The main challenge is to exploit the information owned from the other layers (in particular the routing layer) for avoiding duplicated actions and performing a more efficient service discovery. In the future work, we introduce node without UBR4m registry using the CrossROAD modified as explained in the section 6.1.3.

6.2. A Peer-to-Peer Multicast Application

This section describes the design and implementation of a scalable and flexible whiteboard multicast application (WB). Figure 6.11 shows a snapshot of WB running on two nodes. The whiteboard application provides the basic functionality like subscribing to an arbitrary topic, drawing to a canvas and publishing canvas changes to the associated topic, i.e., sharing the canvas image with the other topic members. Any one node associated to a topic can draw (and share) strokes on the canvas.

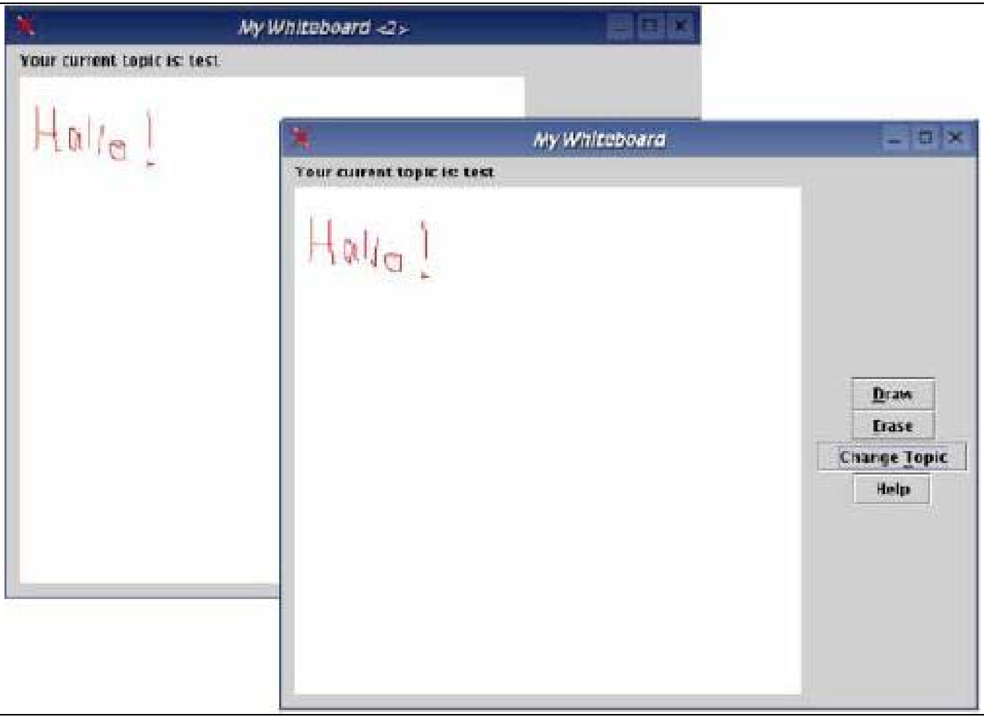


Figure 6.11: Two nodes running the whiteboard application

WB exploits an overlay P2P multicast algorithm as the engine to publish canvas changes to topic members. Scribe is chosen as the multicast algorithm, due to its better performance with respect to other standard (overlay) solutions, such as CAN-Flooding [CJKR03]. Finally, we use Pastry as the overlay substrate for Scribe.

We would like to highlight that Pastry, Scribe and WB are very loosely coupled to each other, thanks to a modular approach, and well-defined interfaces. It is therefore easy to substitute either Pastry or Scribe with versions optimized for ad hoc networks -- e.g., Pastry can be easily substituted by CrossRoad.

WB is a very simple example of group-communication applications, which can be seen as a reference scenario for ad hoc networks. It is thus very interesting investigating how such an application performs in a real ad hoc test-bed.

6.2.1. Pastry, Bamboo and CrossRoad

The original implementation of the whiteboard application assumed Bamboo as the DHT providing the overlay network. Bamboo is a peer-to-peer system primarily written by Sean Rhea of UC Berkeley, but it is based heavily on the OceanStore [KBCC00] and the libasync [ZYDM03] projects. It is written in Java and is available as open source. Bamboo and Pastry shares a similar API, and provide the same kind of services to the above applications.

The management of the overlay network in Pastry is basically "on demand", in the sense that management actions are mostly triggered by nodes joining or leaving the overlay. On the contrary, Bamboo does not react to such triggers, but periodically runs a set of management algorithms to keep the overlay coherent. Bamboo outperforms Pastry in the case of high churn rates, at the cost of increasing the constant load put on the network.
At the time of the first WB implementation, the open version of Pastry (i.e. FreePastry) was much less stable than Bamboo. Therefore we used Bamboo as the overlay substrate of WB and Scribe. However, as FreePastry became more stable, we prefer to port the WB to Pastry. This choice makes WB coherent with the overall implementation of the MobileMAN test-bed. Furthermore, Pastry is preferable because of the lower constant background traffic. Finally, Pastry is more diffused in the literature than Bamboo, and this makes easier to draw comparative evaluations between p2p applications run on wired and ad hoc networks.

It should be pointed out that both Bamboo and Pastry generate quite a lot of background traffic for management purposes. As highlighted by the experiments presented in the Deliverable D8, this should be avoided as much as possible in ad hoc networks. In this view, we believe that using CrossRoad (i.e., using an optimized, cross-layer DHT) would be highly beneficial to the WB performance. Therefore, we are currently porting WB and Scribe to CrossRoad.

6.2.2. Scribe Overview

Scribe is a subject-based, reverse-path forwarding, multicast algorithm implemented on top of an overlay network. For each WB topic a multicast group is defined at the Scribe level. Nodes subscribing for a topic (at the WB level) actually join the corresponding multicast group (at the Scribe level).
Scribe builds, on top of the overlay, a shared tree connecting the nodes subscribed to the same group. Each node may act both as sender and receiver of the group.
Scribe simplifies (with respect to traditional network-level multicast protocols) the join operations and the multicast tree maintenance. Specifically, it leverages the subject-based nature of the underlying overlay network to reduce the maintenance traffic (e.g., flood&prune messages or rendez-vous points dissemination are not required).
Scribe is proved to scale well for large groups of nodes [RKCD02]. Finally, this protocol offers simple APIs to its applications, such as `create(topicID)`, `subscribe(topicID)`, `unsubscribe(topicID)` and `publish(topicID)`. To explain how Scribe works, we now describe two examples: in the first one a new node joins a multicast group, while in the second one a node publishes a message to the group members.

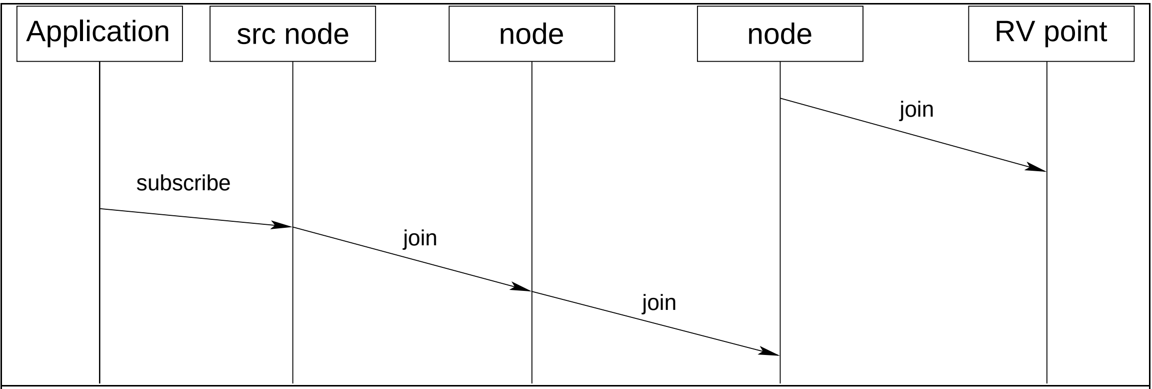


Figure 6.12: A new node joining the Scribe multicast tree

Figure 6.12 depicts the Scribe behavior on a new node joining the tree. For this operation, Scribe exploits a rendez-vous point in the tree. In Scribe the rendez-vous point of the group is the node numerically closest (in the Pastry sense) to the topicID. This is a subject-based way of defining RV points, instead of a topology-based way, as in traditional network-level multicast. Therefore, the RV point can be reached by the standard overlay routing, and no mechanism for publishing the RV point(s) is required, resulting in lower overhead traffic.

In order to join the group, the Scribe module at the joining node sends through Pastry a join message to the rendez-vous point. The first node in the Pastry path towards the rendez-vous point gets the join message and forwards it to the Scribe module. If the Scribe module does not recognize the topic so far (because it is not yet registered to that topic) it discards the join message, creates a new children list for the topic, and adds the source to this list. Then, it tries to join itself to the group by generating a new join message. Eventually, a join message reaches a (Scribe module at a) node that is already member of the multicast group (possibly, the RV point). This node simply adds the source of the join message to its children list, and discards the message. This way, the joining mechanism scales very well, because the join message has just to travel up to the first branching point in the multicast tree (i.e., up to the point where the Pastry path towards the rendez-vous point hits the multicast tree).

Scribe manages the tree as follows. Once in a while (default is 10 seconds) each parent sends a HeartBeat message to each child. If a HearBeat sending fails, the parent assumes that child is no more there, and deletes it from its children list. On the other hand, if a (child) node fails in receiving a predefined number of HeartBeats (default is 2) it assumes to be disconnected from the tree, and hence it tries to re-join.

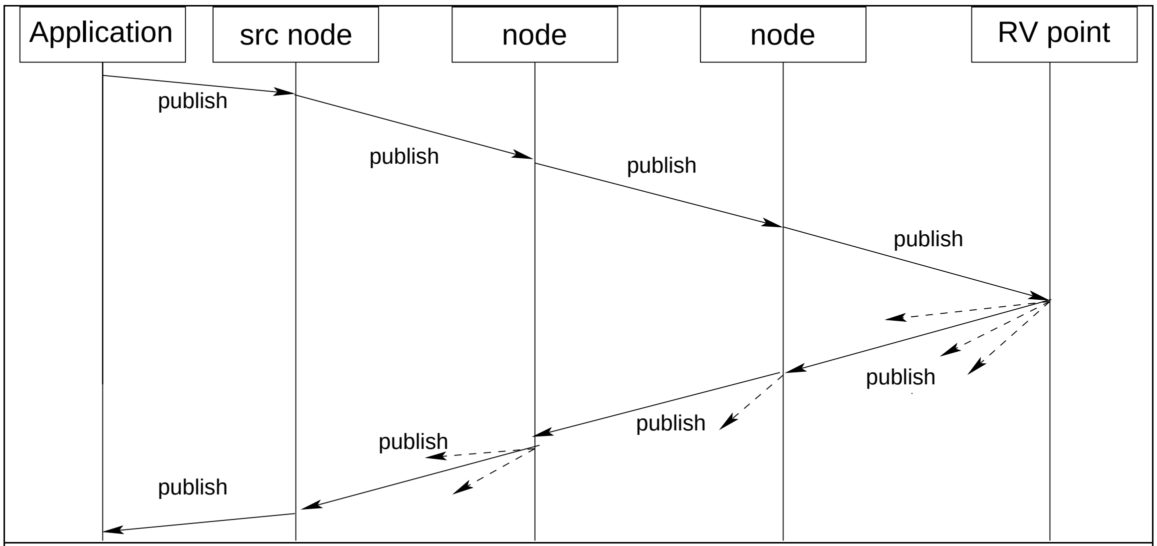


Figure 6.13: Message publishing under Scribe

Figure 6.13 visualizes the mechanisms of the multicast protocol and its interaction with the application when publishing messages. If the Scribe module at the source node would know the RV point's address it could send a message to it directly (like many other DHTs, Pastry provides a way to avoid overlay routing, and sending messages directly to the destination node through network-level routing). Otherwise, the message is routed through Pastry in the usual way. The message reaches the RV point of the multicast tree. Then it is distributed to its children, which in turn forward it to their children, and so on. Any such node checks if there is an application registered for the topic. If so, Scribe forwards the message up in the protocol stack to the application.

A publish message received by the parent implicitly acts as a HeartBeat. Therefore, the counter of missed HeartBeats is cleared upon receiving each publish message from the parent.

6.2.3. The Whiteboard Application

The whiteboard application can be divided into two parts, one is the GUI, and the other is the canvas for drawing. The latter component also manages new data coming from the Scribe, and publishes user strokes through the Scribe.

The GUI is kept very simple, as shown in Figure 6.11, which presents a simple two-node Pastry network using the whiteboard application. At the right side of the canvas are the function buttons for drawing, erasing, switching topic and help. Java Swing extension was used for the graphic implementation. The WB interface is self-explanatory.

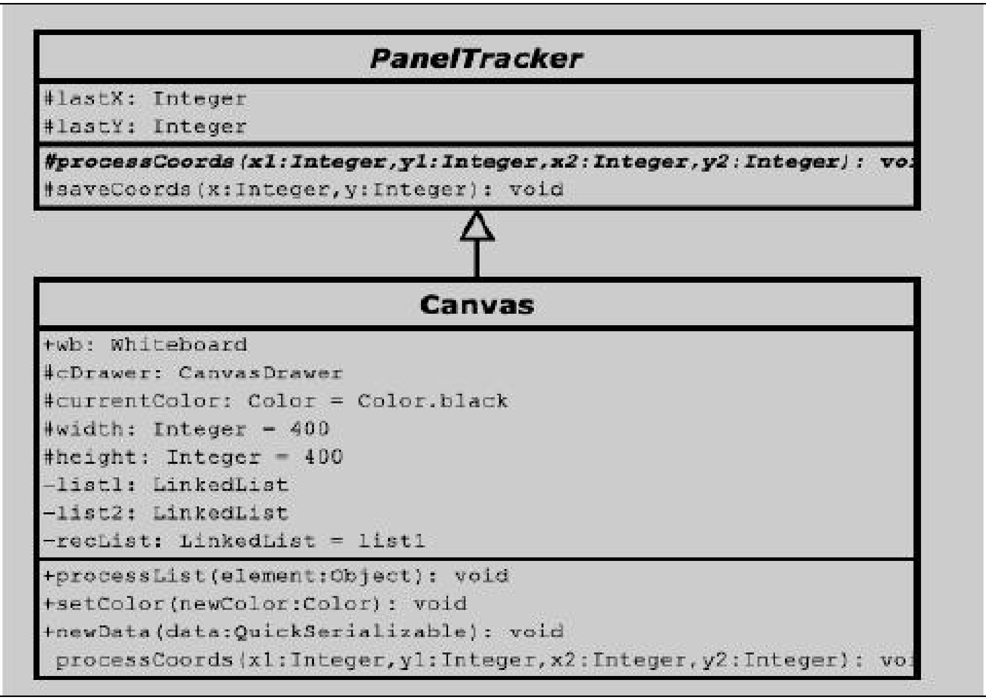

PanelTracker
#lastX: Integer
#lastY: Integer
#processCoords(x1:Integer,y1:Integer,x2:Integer,y2:Integer): vo
#saveCoords(x:Integer,y:Integer): void
Canvas
+wb: Whiteboard
#cDrawer: CanvasDrawer
#currentColor: Color = Color.black
#width: Integer = 400
#height: Integer = 400
-list1: LinkedList
-list2: LinkedList
-recList: LinkedList = list1
+processList(element:Object): void
+setColor(newColor:Color): void
+newData(data:QuickSerializable): void
processCoords(x1:Integer,y1:Integer,x2:Integer,y2:Integer): vo

Figure 6.14: UML diagram of the Canvas class

Figure 6.14 illustrates a slightly simplified UML diagram of the Canvas class. It extends from the `PanelTracker` class which is implemented to manage all user input, such as from keyboard and mouse. Users draw strokes on the canvas with the mouse in the straight forward way. On each mouse movement the last position is saved in the variables `lastX` and `lastY` by calling the function `saveCoords(int x, int y)`. `processCoords()` is invoked on each new line drawing, and stores the new line in a “sending list” for future sending (see below).

Within the Canvas class there are also two inline classes: One for drawing new strokes to the canvas (`DataProcessor`) and the other to send new strokes to the network (`CanvasDrawer`). Both classes extend from `Thread`, and are used to parallelize I/O on the canvas and the main Scribe/Pastry flow of execution. This allows WB to be more responsive both to the user and to the Scribe.

When new data arrive from the Scribe, the whiteboard calls the `newData()` function of the canvas and passes the container with the new data. In order not to block the whiteboard a new thread (`DataProcessor`) is started to process that data and to draw it on the canvas eventually. In order to manage user input, the canvas starts a new `CanvasDrawer` thread at the application start-up. Periodically, the thread wakes up, takes all lines stored by `processCoords()` in the sending list -- if any --, and generates a new message containing all these lines. The message is eventually passed to the Scribe for publishing. The thread then sleeps for half a second.

6.2.4. Simulated Users

In order to have a significant and reproducible testing environment, the user behavior assumed during tests has to be clearly stated. To this end, WB can be used in a "simulated user" mode, where strokes are not drawn by a human user, but instead are generated by a simulated user. The simulated user we have implemented mimics a human user that observes for some time what happens on the canvas, then participates by drawing strokes on it, then observes again, and so on. To this end, the simulated user behaves as follows: it sleeps for a random amount of time, then it draws a random number of lines (i.e., a burst) and goes back to sleep. The experiment finishes after *N* such cycles, where *N* is a simulator parameter. Both the burst size and the sleeping duration are exponentially distributed. The average values of these distributions are simulator parameters.

6.2.5. Performance Figures

We are currently running experiments to assess the WB and Scribe performance over ad hoc networks, and understating to what extent cross-layering (i.e., CrossRoad) can help in this contest.

As a first step we define the performance figures for our analysis. We define two classes of indexes. The first-class indexes aim at quantifying the "User QoS", i.e., the user satisfaction in using WB. We consider two parameters in this regard, i.e.: i) the delay experienced by receivers to get source nodes' strokes, and ii) the percentage of messages that are lost. We believe that this couple of indexes are able to capture the satisfaction of the WB user. Therefore, they are valid tools to evaluate if the legacy ad hoc network stack is suitable for supporting such kind of applications, and to what extent cross-layer optimizations are beneficial to the user QoS. At a higher level, these indexes will indicate if such kind of group-communication applications are suitable for ad hoc networks.

The second class of indexes is devoted to measure the efficiency of the multicast tree built by Scribe. To this end, we define indexes pretty similar to those used in the literature to evaluate overlay multicast algorithms. A first index is the link stress, i.e., the number of messages sent on a physical link for each message generated by the application. This index shows the overhead in terms of additional messages due to building the multicast tree at the overlay network instead of at the network level. A second index is the node stress, i.e., the number of children a node has in the multicast tree. This index shows how well the load is balanced among nodes in the tree.

It should be noted that in an ad hoc network the multicast tree will probably be quite dynamic -- also when nodes do not move -- due to link breakages, high delays, congestion etc (see results in Deliverable D8). It will be thus interesting monitoring the above indexes over time -- especially the node-stress index, and showing their variability during the experiments. Of course, the higher the variability, the higher the overhead traffic. Specifically, each time a node looses its parent, it has to join again to the tree by following the procedure described in Section 6.2.2.

In this regard, to assess the frequency of variations in the tree structure, we also monitor the parentID of each node over time. Furthermore, we define the "re-join index", as the ratio between the number of joins a node must send (after the initial one), and the number of messages get from the parent. This index shows how frequently a node has to look for a new parent, due to a disconnection from the previous one.

6.2.6. Conclusion and Future Works

The development of the whiteboard multicast application provides a simple but effective application example for the MobileMAN test-bed. Due to the flexibility of the design, both application and multicast modules could be reused easily. Thanks to a modular design, WB can be used both in the traditional protocol stack, and in the optimized cross-layer stack, thus showing the benefit – *for the end user* – of cross-layer architecture. Furthermore, a lot of other applications, such as video streaming, could be developed based on the same multicast layer created.

We are currently evaluating the performance of WB and Scribe both on a traditional ad hoc network stack, and on an optimized, cross-layer stack. We envisage significant advantages in integrating this application in the MobileMAN cross-layering architecture. This architecture makes the context from other layers, such as the routing information, available to the middleware layer. For example, this would be beneficial to the optimization of the multicast tree in Scribe. We are currently investigating this research direction.
This work also provokes another interesting research point: use of the peer-to-peer multicast to interconnect IP multicast networks by tunneling non-multicast capable networks.

6.3. Real time audio: VoIP

The Voice over IP is a real time application that is quite demanding for Ad Hoc networks. However, voice communications is a service that will provide added value to Ad Hoc networks since users will benefit from a communications without infrastructure support.

Following the same criteria and trying to re-use existing implementations we faced several problems when considering devices with limited resources such as PDA (i.e. iPAQ). The existing implementations of VoIP services required devices with enough processing power. Therefore, in order to develop a VoIP service for real scenarios including portable devices with low resources, we had to implement light version of VoIP application.

The VoIP application contains two main modules; signaling module and data transport module. The results show that a VoIP service can be provided in Ad Hoc networks with reasonable quality. However, since users have a good VoIP service with fixed networks the existing implementation requires a QoS module in order to enhance the service quality and meet user expectation.

6.3.1. Signaling module

The signaling module consists of the software component that will initiate the VoIP session with other peer nodes in the Ad Hoc network. This module has been implemented specifically for the Ad Hoc framework since existing implementations did require excessive resources (e.g. CPU, memory, etc). The signaling module implements the SIP signaling protocol and utilizes IP addresses for finding the peer nodes to initiate the VoIP session. The SIP signaling protocol can run on UDP or TCP protocol but in order to minimize the requirements for maintaining the session state in the nodes, the existing implementation uses UDP as the only transport protocol. The session initiation also requires negotiating the media parameters using SDP protocol. In order to minimize the negotiation process the signaling module uses the same codec for the VoIP session (i.e. GSM). Therefore, the signaling module is compliant with the SIP protocol but having a single codec optimizes the session set-up.

The SIP module is implemented specifically for the Ad Hoc network but the GSM codec is obtained from public source [].

6.3.2. Data transport module

The data transport module consists of the software component that after the VoIP session is set up, takes care of exchanging the voice packets coded with the selected media format (i.e. in this case GSM is the only codec used in the session).

The data transport module implements a RTP client for exchanging the voice packets. The RTP client implements the functions for obtaining the audio samples from the microphone, encoding them using the selected codec (i.e. GSM) and then exchange the packets using the RTP protocol. The RTP client uses a publicly available RTP library for managing the RTP messages [12].

6.3.3. VoIP testing on Ad Hoc Networks

The VoIP application implemented was tested at CNR (Italian National Research Council) in Pisa on 30th September and 1st October 2004. Results of this testing are documented in Deliverable D10. These tests pointed out quality of service problems partially due to buffer management schemes in the end-to-end nodes running the applications. For this reason during the first part of the third year the application software was refined. Preliminary testing results performed on mid-May in CNR campus pointed out a significant improve in the quality of service. More extensive test are currently ongoing to confirm these preliminary results.

6.4. References

[CGMT04]	M. Conti, S. Giordano, G. Maselli, G. Turi,: "Cross-Layering in Mobile Ad Hoc Network Design". IEEE Computer 37(2): 48-51 (2004)
[FP]	FreePastry web site. http://freepastry.rice.edu
[G99]	Guttman, E. (1999). "Service Location Protocol:Automatic Discovery of IP Network Services." Internet Computing(July-August): 71-80
[GN03]	Guttman, E., T. Narten, et al. (2003). Zero Configuration Networking, IETF
[IF03]	Iyer, B., J. Freedman, et al. (2003). "Web Services 1: Enabling Dynamic Business Networks." CAIS
[KBCC00]	J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells und B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, Banff, Canada, Nov 2000.
[KF02]	Kindberg, T. and A. Fox (2002). "System Software for Ubiquitous Computing." IEEE Pervasive Computing (January): 70-81
[MM]	MobileMAN Project, http://cnd.iit.cnr.it/mobileMAN
[RD01]	Rowstron and P. Druschel – Pastry : Scalable, decentralized object location and routing for large-scale peer-to-peer system, Middleware 2001, November 2001
[RGRK03]	Sean C. Rhea, Dennis Geels, Timothy Roscoe und John Kubiatowicz. Handling Churn in a DHT. Technischer Bericht UCB//CSD-03-1299, University of California, Berkely and Intel Research, Berkely, Dec 2003.
[Rhea04a]	Sean Rhea. Bamboo: Programmer's guide.http://www.bamboo-dht.org/programmers-guide.html, 2004.
[RKCD02]	Antony Rowstron, Anne-Marie Kermarrec, Miguel Castro und Peter Druschel. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)* 20(8), Oct 2002.
[SMKKB01]	AI.Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", SIGCOMM'01, August 2001
[UD02]	UDDI Version 2.03 Data Structures Reference, UDDI Committee Specification, 19 july 2002.
[WeCB01]	Matt Welsh, David Culler und Eric Brewer. SEDA: An architecture for well-conditioned, scalable internet services. In *Proceedings of the Eighteenth Symposium on Operating Systems Principles (SOSP-18)*,Oct 2001.
[ZYDM03]	Nickolai Zeldovich, Alexander Yip, Frank Dabek, Robert T. Morris, David Mazi`eres und M. Frans Kaashoek. Multiprocessor support for event-driven programs. In *Proceedings of the 2003 USENIX Technical Conference,* San Antonio, Texas, Jun 2003. S. 239–252, section 2.
[D05]	F. Delmastro, "From Pastry to CrossROAD: Cross-layer Ring Overlay for AD hoc networks", Proc. Mobile Peer-to-Peer 2005 in conjunction with PerCom2005 Conference, Kauai Island, Hawaii, March 8-12 2005.
[CJKR03]	M. Castro, M.B.Jones, A.-M. Kermarrec and A. Rowstron, "An Evaluation

	of Scalable Application-level Multicast Built Using Peer-to-peer Overlays", Proc. of INFOCOM 2003.
[1]	GSM codec library, http://kbs.cs.tu-berlin.de/~jutta/toast.html
[12]	JRTPLib version 2.9, http://research.edm.luc.ac.be/jori/jrtplib/jrtplib_old.html

7. ECONOMIC MODELS

As the interest in mobile networking increases amongst people within the technical and commercial sectors, there is a strong drive to develop solutions that will meet the needs of the community while at the same time provide business incentives for corporations to become involved in making these networks possible. Many types of mobile networking solutions are being proposed and tested, including enhancements of cellular networks such as 3G networks, the proliferation of WiFi hotspots and mobile ad hoc networks. Each solution aims to provide high bandwidth access to mobile users, while facing unique technical challenges before the network can be deployed and good coverage achieved. At the same time, the networking community has learnt through hard experience that well-designed business models need to accompany good technical solutions so that the probability of overall success can be raised.

Where technical solutions seek to develop protocols that allow devices to communicate reliably within a network, business models take a top-level view and consider all the players that need to come together to create an operational network. These models determine the relationships that exist between players and how these players can influence outcomes for the network. To get a concrete understanding of the issues involved, usage scenarios can be envisaged revealing some of the situations that will be faced during the operation of the network.

Hereafter, we are specifically concerned with the development of business models for the MobileMAN project. The MobileMAN project aims to demonstrate that a fully functioning, medium size mobile ad hoc network can operate effectively within a metropolitan area. Mobile ad hoc networks are unique in the sense that the network is formed through the cooperation of individual users, where their devices perform the forwarding that is necessary to achieve network capability. As a result, there is no network operator as such, which means that standard business models for networks cannot be directly applied. Even hybrid scenarios that incorporate access points into ad hoc networks are considerably different as the carrier is no longer a dominant entity. Our goal is to develop a business model specifically for networks that will be developed within the MobileMAN project [20], and to identify key business issues that need to be addressed.

7.1. Business case studies for ad hoc users and their applications

Firstly, we introduce a framework which allows us to classify typical business players that are required to deploy and operate general networks. Since the MobileMAN project has specific target groups within the community, this naturally leads us to presenting usage scenarios. These scenarios provide us the opportunity to highlight key needs of users. We then proceed to analyze the business opportunities for networks such as those proposed in MobileMAN in more depth, and we make key recommendations for the direction of the project with regards to achieving financially viable networks.

7.1.1. A Conceptual Framework for Business Models

Before we launch into detailed descriptions of the objectives of the MobileMAN project and also of the possible usage scenarios that will enable us to develop business models for mobile ad hoc networks, a conceptual framework will be introduced which can be used to classify the major components involved within a business model for such a network [3]. By clearly identifying each component, we can consider the actors who will accomplish the necessary functions in order to meet the various needs of the users of the network. In fact, there can be many players which need to come together to create a working mobile ad hoc network, and these players include device manufacturers, software companies and government regulatory bodies, not to mention consumers themselves who are willing to join and cooperate within such a network. The fundamental contributions and relationships of each player can be identified, allowing the players to be classified and incorporated into a single framework as shown in Figure 7.1. This model provides a structured way of the thinking about the system and a meaningful point of departure to build a business model.

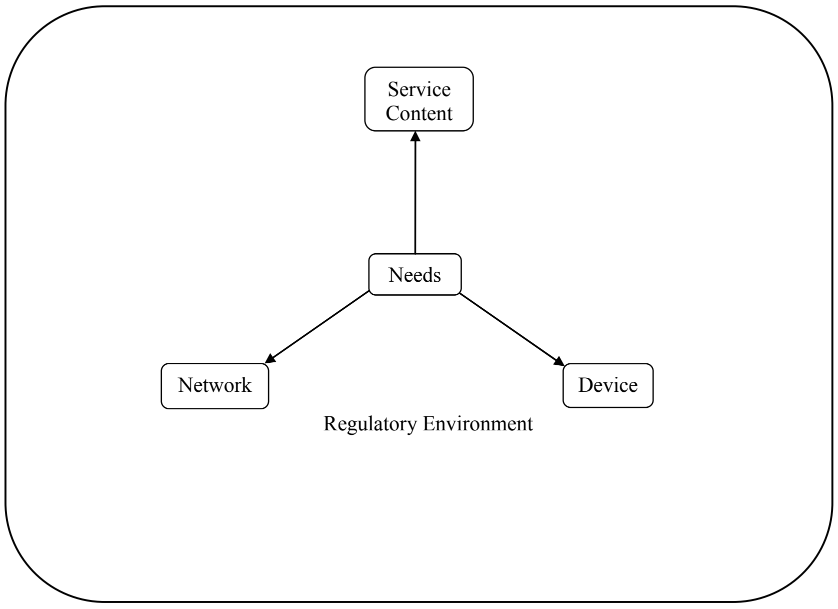


Figure 7.1. Classification framework for the business model

The needs of network users lie at the centre of the framework and define the purpose to which all the other components in the model should focus on. These needs can encompass a wide range of aspects, including the application needs of various users, whether the network is required for an organization or for a collection of individual users, and if the users are mobile during their use of the network. The first phase of developing a business model is to clearly identify what the needs of the users are. While there may be detailed needs that are specific to individual users or that change over time, the model should identify the essential requirements of the users in general.

There are three main categories of players who focus on meeting the needs of network users. The first of these are players associated with the actual devices. These players include the manufacturers of the devices and retailers of the devices. For a mobile ad hoc network of individual users, devices within the network would only include portable devices such as laptops, PDAs and cellular phones, while a network within a factory may include other devices such as sensors and bar code scanners. Operating systems are also an important component of functioning mobile devices, so companies such as Microsoft would play an important role in business models for mobile ad hoc networks. Finally, if a mobile ad hoc network is associated with fixed access points at any point during its operation, then these nodes would also be devices associated with the network.

Service and content is the next category that is associated with a network. Mobile devices may have many stand-alone software applications, such as electronic diaries and address books, office software and individual games. These applications can certainly be considered within the business model for mobile ad hoc networks. However, the model is primarily concerned with networking applications. These include basic applications such as messaging services and device/user location software to more sophisticated applications such as gaming, auctions and group management systems. Players that are involved with the development of these types of applications need to be considered within the design of the business model.

Within traditional business models for telecommunications, companies involved in the development and deployment of any networking infrastructure form another major component. For self-organizing networks, such as mobile ad hoc networks, networking capability is embedded in the networking protocols within each device rather than being provided by a specific player in the model. Thus, the business benefits for a purely ad hoc network are more difficult to identify. However, whenever a connection is formed between a mobile ad hoc network with either an operator-driven network, such as a GSM or UMTS network, or a wireless LAN, then more obvious business opportunities arise. In this report, we will describe various situations when these opportunities occur.

Finally, all of the players within each category of the business model must work within a regulatory environment, which includes spectrum access and management, customer rights with regards to devices and applications, and other regulatory bodies. In developing business models for MobileMAN, we will highlight any situations where the regulatory environment has a significant role in determining the outcome.

7.1.2. Evaluation of MANET's

In order to build strong business models for MANET, it is first necessary to identify its core strengths and weaknesses in order to come up with scenarios and applications best suited to it and identify the user segments most likely to benefit.

Strengths:

MANETs are highly conducive to rapid, innovative developments. Since no infrastructure needs to be installed, there is zero deployment time required.
Provides flexible inexpensive networking solutions built around co-operative interactions amongst users, supporting free information sharing.
It is a low cost, relatively equal opportunity method of networking. Population or geographical segments that would have been unprofitable and hence underserved or ignored by corporations can be connected together at only the cost of the devices themselves.
In situations where it is not financially or functionally possible to provide an administrator to run and maintain a network, MANETs are very useful since they are self organized.
MANETs are potentially very mobile; as long as there are nodes to act as intermediates, coverage is available. This is in contrast to wireless LANs for example, which requires its users to remain within a certain coverage area.
MANETs are also more robust than other forms of networks since their performance is not subject to electrical outages or damaged infrastructure.

Weaknesses:

Both the technology and concept are unproven, and potentially there could be a significant Quality of Service (QoS) issue, especially if users group together in certain areas leaving gaps where intermediate nodes are needed. Human movement tends to be individualistic yet certain

overall patterns can still be found. Large groups of people congregating or leaving an area at the same time are not uncommon; for example, a huge exodus of people from the city during rush hour. These conditions may result in sudden outages or areas with periodically poor coverage. On the flipside, there may also be interference issues if too many users congregate in one area.
For many applications, such as for a wireless network for an office, it would make more sense to simply implement a wireless LAN system as the additional cost would not be significant and well worth the additional performance.
While the upside is that administrators are not needed for MANETs, the downside of that is that there is also no control, regulation or support for the system when things go wrong.
The purely ad hoc scenario faces severe limitations in not being able to interact with other networks, most notably the Internet. It is therefore restricted to users who only wish to interact and share information between them. This limitation is of course removed when access points are added.
The cooperative requirements of MANETs may make it unsuitable for general applications where users might wish to game the system for their own benefit, unless suitable preventive or incentive measures can be devised. This situation is exacerbated by the fact that there is no centralized governance.

From the analysis above, it becomes apparent that despite a number of distinct advantages, MANETs will only be compellingly useful in certain scenarios, especially in the purely ad-hoc case, owing to the outlined limitations. Generally speaking MANETs will be most useful when one or more of the following is true,

The objective is to connect communities of people with similar interests or goals, who would not otherwise be connected for whatever reasons.
The objective is to enable wireless networks in areas where centralized infrastructures are impossible, undesirable or unnecessary.
Predictable quality of service is not of paramount importance.
Mobility at a low cost is important.

Given these considerations, it follows that from a practical business viewpoint, there will be three main classes of ad hoc networks, specifically, purely ad hoc networks which are completely decentralized and have no access points, ad hoc networks which are connected to centralized servers that provide specific information or services, but without conventional access points (effectively a central ad hoc terminal that talks to all other ad hoc terminals in the network) and finally ad hoc networks which are linked to the wider Internet through access points.

By using these hybrid models, the true value of ad hoc networks can be determined, especially relative to a regular wireless LAN network. An ad hoc device might therefore need to have three operational modes corresponding to the three classes, depending on where the users are and what they are doing at that point in time.

It is also possible that ad hoc devices could be designed to include location finding capabilities by including synchronization clocks and using signal power measurements, although it would probably not be as accurate as a GPS system. This would open the possibility of additional applications in all three classes.

7.1.3. Examples of Potential Applications

In this section a few potential applications will be detailed. In the next section a further three applications will be fleshed out in more specific scenarios to give a greater understanding of how these applications might work, and to show what the respective business models might look like.

As mentioned in the preceding section, amongst the most compelling advantage of purely ad hoc networks is that they have zero deployment times and enable wireless networks where centralized infrastructure is impossible, undesirable or unnecessary. This leads to potential applications such as,

Emergency services. Although in most situations, emergency services might not wish to rely on ad hoc networks due to its unpredictable quality of service, in a disaster type situation such as a flood, an earthquake or a severe power outage, base stations might easily be destroyed, damaged or otherwise rendered inoperable. In this sort of situation an ad hoc network would be instantly deployable and immensely useful for communications and information sharing in rescue operations.

Scientific expeditions, school field trips, wildlife adventure treks and other such trips to remote areas would benefit greatly from ad hoc networks since users would be able to communicate wirelessly in areas which would otherwise have no coverage at all. A group could share information and use instant messaging services wherever they are, be it the artic circle or desert wasteland, enabling the sharing of discoveries. If one or more of the ad hoc devices were connected to a GPS network, it could be even more useful, allowing group members to share location information with each other.

Homes for the elderly and the handicapped could benefit from an ad hoc network in cases where it is not financially or functionally feasible to have a centralized wireless LAN system. This would allow residents to communicate, share information and play games with each other wherever they might be, even if some of them were not mobile enough to move around easily despite being within the same residence. The main benefits in this application would be much lower (free) running costs, compared to the wireless LAN case where a system administrator needs to run and maintain a centralized base station and server.

7.1.4. Usage Scenarios

Following on from the previous sections, we designed three scenarios to illustrate the wide variety of possible uses possible using ad hoc networks. These scenarios show that applications for ad hoc networks need to be tailored to take into account the unique strengths and weaknesses of the technology as well as the fact that unique business models need to be used to make the networks financially viable. They also highlight the problems that are likely to be faced when implementing such systems.

City Cabs Scenario

One of the most important components of a city cab company's operations is its dispatch unit. The dispatch unit informs individual cabs about passenger pickups, assigns passengers to nearby empty cabs, and passes on any additional information that may be necessary to drivers such as directions and news about weather or traffic conditions.

Traditionally, cab companies have relied on radio dispatchers for these tasks. Radio dispatch systems have variable QoS issues, and there are significant costs associated with installing, running and maintaining them. In most cases, radio licenses need to be obtained to operate these systems, adding additional cost and creating barriers to entry for small companies. Economies of scale mean that large companies are able to share the cost of radio dispatch amongst many drivers, whilst smaller players are unable to afford these systems.

However, the popularization of mobile phones now means that smaller companies are able to get into the picture by relying on the public mobile network as their dispatch system. Unfortunately, this method has many drawbacks including cost and the fact that dispatchers can currently only communicate with one driver at a time and direct communication between drivers is limited and less coordinated. Voice relayed instructions are also subject to higher error rates.

Many bigger companies have in turn upgraded their own systems to include smart software with GPS which allows smart tracking of cabs and efficient allocation of jobs, improving productivity and utilization of resources. However the hardware and software required is costly (costing over $1.15 million for a company with 300 taxis [10]) and its operation involves transferring data wirelessly over public or private mobile radio networks, which is a significant recurring cost.

In our scenario, an ad hoc system might be used, with each cab fitted with an ad hoc device and a central ad hoc server at dispatch headquarters. As long as there are enough cabs around the city to form a network, this ad hoc system could be used to efficiently transmit dispatch information. Hardware costs would decrease significantly as only low maintenance ad hoc devices would be required. Operational costs would also be low since the cost of transmitting data would effectively be zero, compared to the heavy traffic costs incurred by dispatch using mobile phone systems and some radio systems.

How it might work

As usual, a customer would call in to the cab company's dispatch headquarters. The call would then be picked up by a dispatch handler who would enter the relevant details of the job into the computer system. An automated request would then be sent via the ad hoc network to all free cabs in the vicinity. In the case of ad hoc devices enabled with location information, the nearest free cab to the pick up point could be automatically located and assigned the job, thereby decreasing waiting times for the customer and improving turnover rates for the cab company. The ad hoc system would constantly be updating itself automatically with relevant job and road condition information for drivers, and drivers would also be able to communicate amongst themselves, empowering all parties concerned with the information they need.

Taxi stands throughout the city would also be equipped with ad hoc terminals that automatically route cabs to appropriate stands when requested. Should ad hoc devices become ubiquitous, it would even become possible for customers to simply order a cab via their own standard ad hoc devices wherever they were and have the request sent directly to the nearest cab. The cab would then register the pickup and send the information through the network to update his counterparts and headquarters of his status. With the addition of an access point at dispatch headquarters, customers would also be able to book cabs over the Internet and have the request relayed immediately via the ad hoc network.

In all the cases mentioned previously, there would a two way flow of information; once a particular cab driver has accepted a job, details such as the cab's registration number, its current location and the estimated time of arrival can be sent back via the ad hoc network to the user's phone via sms, to the taxi kiosk, to the customer's ad hoc device or to the messaging service of the Internet customer respectively, reducing any potential misunderstandings that are common with voice relayed radio dispatching services and allowing better planning on the customer's part.

If the network were secure enough, credit card payment options could also be implemented using the ad hoc terminals, with encrypted verification information sent to and from the central node, providing more convenience to the customer.

The value chain shown in Figure 7.2 is a map of the primary players within the cab company radio dispatch industry. These players make contracts between each other to conduct exchanges of information, money, services or a combination of the three. Dotted lines indicate the flow of information or services, for example the information that is shared between cab drivers and dispatchers, whereas solid lines indicate a monetary exchange, for example when a customer pays a cab driver for the cab ride. The value chain map allows the relationship between the major players to be established at a glance and allows a better understanding of where value is or needs to be generated.

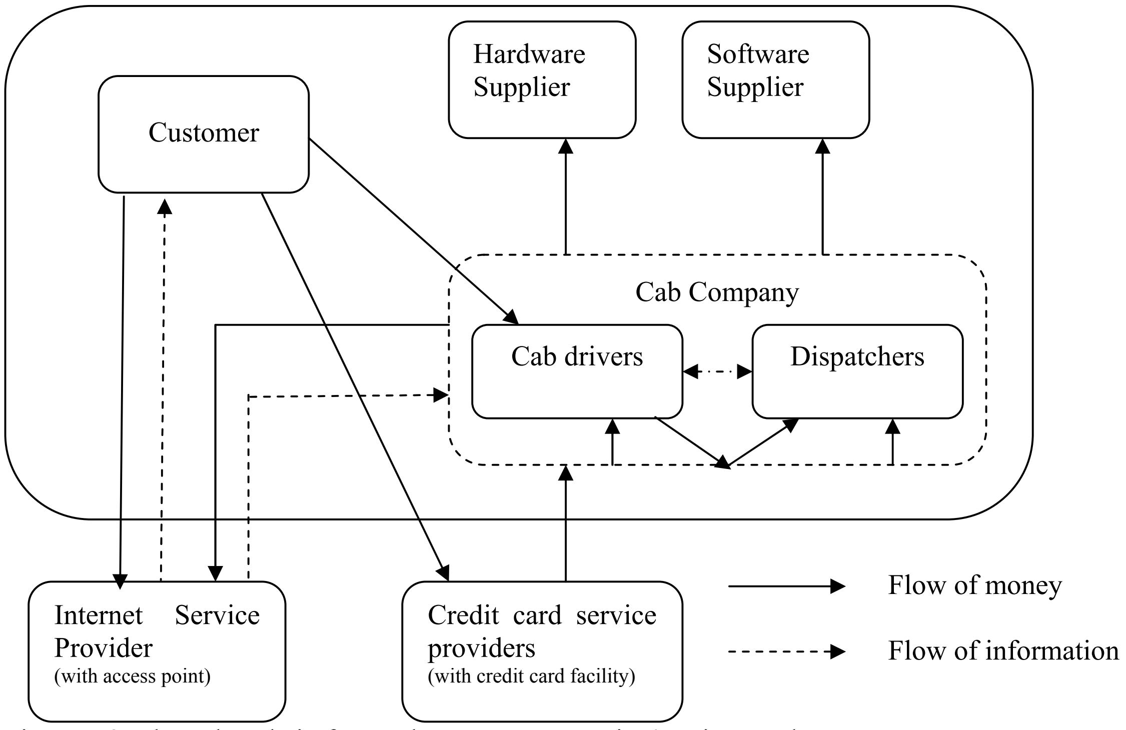


Figure 7.2. The value chain for a cab company operation's primary players

Value proposition for primary players involved

For customers:

1. Bookings can be made with greater ease and convenience through a variety of different mediums besides the phone, and time spent on making bookings can be reduced.
2. The information flow is two way and customers can use the information provided to reduce unnecessary time spent waiting for a cab that may only be arriving some time later or looking for the correct cab in a crowd of look-alikes.
3. With location specific job assignments and less information congestion on the dispatch side, cabs can get to customers much faster, reducing their overall waiting times. The chances of getting a cab should also be improved due to the higher turnover rate and accuracy of the system.

4. Customers will be able to enjoy a quieter environment in cabs, without constant radio chatter from dispatch headquarters.

For cab drivers:

1. Faster and more efficient processing of jobs means that drivers will be able to experience faster turnover rates and have less empty cruising time, thereby increasing revenue and profits.
2. Decreasing reliance on voice dispatching results in fewer misunderstandings between dispatchers and drivers.
3. With location information and a hidden alarm switch, drivers in distress can immediately be located and helped, in the event of an accident, a mugging, a carjacking etc. [9]
4. Cab drivers will be able to enjoy a quieter environment in cabs, without constant radio chatter from dispatch headquarters.

For cab companies:

1. Streamlining the order taking, order processing and dispatching process should result in significant capacity increases, increasing both revenue streams and service levels at the same or lower costs.
2. An ad hoc system will result in significant cost savings, especially in the long term as operational costs will be marginal compared to conventional methods.
3. High customer satisfaction and employee satisfaction will result in enhanced reputation and hence demand for the cab company's services.
4. With better real time information flow, management's decision making capabilities will be enhanced, and the reaction time for a decision to filter down to the driver level will be decreased.
5. Location information allows cabs to be tracked in the case of emergencies, as well as to allow better route forecasting and resource allocation.
6. Cab companies have limited opportunities for differentiation, especially in areas where prices are fixed by law; product differentiation is limited to fast, courteous service, comfortable vehicles and perhaps most importantly, convenience of booking/flagging. In such highly competitive, low differentiation businesses it is very important to avoid making mistakes that will stick in customers' minds (generating bad word of mouth and causing lost repeat business). Ad hoc networks as outlined previously not only make it more convenient for customers to book cabs, but they will also minimize mistakes made through driver-dispatcher misunderstandings.
7. Ad hoc systems are more robust than centralized systems, and service is not dependent on service providers or subject to damaged infrastructure.

To taxi dispatchers:

1. As customers get used to alternatives to booking over the phone, the workload for dispatchers will decrease.
2. Reliable automated systems will improve the accuracy and speed with which orders are taken and passed on.
3. By sending booking information accurately over the network, misunderstandings over voice instructions will be minimized, decreasing conflict in the workplace.

Risks & Disadvantages

Despite all the potential benefits that an ad hoc network poses in this scenario, there are also a number of potential risks and drawbacks to the system that must be noted and either pre-empted or overcome.

Coverage issues:

By the nature of being an ad hoc system, good coverage depends on having a suitably spread out number of nodes around the required communication area. This means that to work properly, there must first of all be a sufficient number of cabs to cover the entire service area.

If too many cabs are sent to the outskirts of the cabs' service area at any given time, there is a danger that there will be gaps where there are insufficient nodes to act as relay points, hence causing a breakdown in communications.

Similarly, if too many cabs happen to group around a certain area (e.g. to pick up a large number of customers after a big football match) it is possible that interference may disrupt communications.

There is also the problem where even though a reasonably sized cab company will probably have cabs covering most parts of the city; it is highly probable that at least some customers may wish to make longer journeys out of the city. That would certainly take the cabs out of the ad hoc coverage area (unless we are at a later stage of the roadmap, where ad hoc devices have become ubiquitous and nodes other than the cab driver's devices may be used as relay points) and hence render that particular cab incommunicable.

Although the above disruptions may only be momentary in nature, the unpredictability of QoS is a significant risk to cab companies who rely on the network to serve customers. The lost opportunity cost of losing customers by being unreachable is not only significant financially, but also in terms of brand name reputation and may well overshadow the cost advantages of implementing an ad hoc network in the first place.

In order to mitigate these problems, a number of steps could be taken. First of all, to ensure satisfactory coverage, stationary ad hoc relay points could be placed at strategic points throughout the city, so that even if no cabs are in the vicinity there are still nodes that can relay information. Ideally, these stationary relay points could also double as taxi stands.

Secondly, backup systems should be prepared such as mobile phones so that cabs can still be contacted should there be a network split or if a certain cab is serving a customer who wishes to travel outside the usual coverage area.

Finally, cab companies should attempt to standardize their ad hoc devices and cooperate with other cab companies so that their ad hoc systems can use each other as relay points, even if the software and information they send are themselves incompatible.

Security issues:

The transmission technology used must be sufficiently secure such that sensitive information is not intercepted. If there were a standardized ad hoc network between different cab companies, each company would need to make sure that their own information was encrypted safely, to avoid potential problems where the drivers of one company attempt to poach the customers of another. The security issue is even more important when credit card processing facilities are offered to customers.

Scalability issues:

The entire system must be scalable to avoid potential problems in the future. In the case of a standardized system used between companies, this means that both the software and the communications technology must allow significant growth of the number of ad hoc devices in use. However, with restricted bandwidth in a small area, there is always a limit to the number of users that can be supported, and therefore should the technology become truly successful, it may inherently cause problems with the effectiveness of the system. It is unlikely that this would be the case for cab companies alone, but in the scenario having ubiquitous ad hoc devices in a metropolitan area, there is a significant risk that problems would occur. This is a technological and regulatory issue that needs to be solved with a two-pronged approach, by both improving the technology and by regulation, ensuring that no bandwidth is overused by too large a group at any one time.

Substitutes and alternatives to ad hoc:

The main alternatives to ad hoc systems are communication systems that run off of satellites, public networks or proprietary private networks. Research is also being performed to link together conventional wireless LAN systems such that handoff can be achieved seamlessly between them. The main advantages that ad hoc networks have over these other forms of communication are low cost and ease of deployment. Should the alternatives advance sufficiently in performance or decrease sufficiently in price, they could render ad hoc systems inadequate in comparison.

Cost issues:

Although an ad hoc system will allow companies to save significant costs for wireless communications, there are still costs associated with the purchase and maintenance of devices, the customized software required to run the system and training costs for drivers and dispatchers who will need to use the system.

It is likely that as ad hoc devices become more popular and production volume is increased, prices for ad hoc devices will be driven steadily downwards. Given the standardization of the components required and the technology standards in place, it is unlikely that hardware sellers will be able to exert much power in terms of pricing as there will be significant competition for sales.

Software vendors will be in a slightly better position to exert pricing pressure, however despite the customized nature of the software required, there are already a large number of vendors catering to cab companies, albeit for systems that communicate using private or public wireless networks. However the basic nature of the software remains the same and should be easily adaptable to the ad hoc case.

In light of the risks detailed in the rest of this section, a detailed cost benefit analysis should be done to ensure that the long-term benefits outweigh all aspects of potential risk and cost involved.

Multiplayer Gaming Scenario

Market Potential

The past decade has seen tremendous changes in the computer/video gaming industry. From humble beginnings with a few small developers setting out to create toys for teenage boys, the gaming industry is now a multibillion dollar industry, valued between $30 and $35 billion in 2002 [17] (larger than the motion picture industry in terms of box office revenue), offering social entertainment that appeals to many different demographics (over 60% of gamers are now over the age of eighteen).

Multiplayer gaming in particular has gained massive popularity in the past few years, being a natural extension of gaming with the added dimension of social interaction and competition. The main enabler for this growth has been the continual improvement in the price-performance of processing speed, graphics, and bandwidth to satisfy the huge demands that games require.
The market for games continues to expand every year. At the moment it is estimated that more than 200 million people worldwide play PC games, more than 140 million play console games and another 100 million play handheld games [17]. Online gaming is a rapidly emerging segment, enabling multiplayer gaming predominantly through PCs and console devices.
Wireless gaming is still at an infant stage and at present mobile devices mainly support simple Java games that are downloaded from the operator networks and do not tend to have extensive multiplayer capabilities. These games currently represent less than 1% of the overall market according to Informa Global Videogame Market. This figure is estimated to grow to 12% by 2006. Analysts IDC report that there were 7 million wireless gamers in 2002, and they expect that number to increase to 71.2 million in 2007 [20].

Already there are concerted efforts being made to secure a foothold in this market segment. Partnerships are being formed between many wireless network operators or device manufacturers and video games giants, including collaborations between Motorola and Sega, Nokia and Eidos, Orange and Rage, and NTT and Nintendo.

Nokia has recently released its Ngage system, which is a mobile phone designed with the dual purpose of being a handheld gaming device. However, at present it offers only one multiplayer game for download over WAP, which is relatively simple in nature and does not support many players at once. Along the same lines, Cybiko is a proprietary wireless device aimed at teenagers which allows multiplayer gaming. However, its games tend to be in black and white only and its range is limited to less than 300 feet.

Lifestyle changes in the main target demographic of 16-30 year olds will drive the importance of mobility in the games market. As this demographic grows increasingly accustomed to mobile consumer electronic devices that permeate all aspects of their active, mobile lifestyles, they will desire the ability to perform more and more functions (including playing games) on the move.

It can therefore be concluded from the preceding points that since,

the overall market for games is very large and growing,
there is a distinct trend towards increasing mobility,
multiplayer games are becoming very popular,

there is huge potential to be tapped in the still nascent mobile multiplayer games market.

Multiplayer games

The appeal of multiplayer games is twofold. Firstly, the ability to play with other people completely changes the gaming dynamic from being a solitary activity to becoming a social activity that allows you to play with existing friends or make new ones, instantly making it more appealing to a large number of people. Secondly, for gaming enthusiasts, real human opponents are more challenging and motivating than any computer generated opponent.

There are two main categories of multiplayer games: Small Group Multiplayer games and Massively Multiplayer Games.

Small Group Multiplayer games are usually limited to less than 32 players at any one time in any particular game. Within this category there are in turn many different genres of game, including real time strategy, first person shoot 'em ups, fighting, simulation, racing and sports. These games usually require all participating players to be present before a game can start.

Other types of Small Group Multiplayer games include board, casino, card and simple arcade games. These games are relatively simple, graphically less intensive and communication amongst players is usually quick and minimal. Players tend to number eight or below at a time. In some cases, players do not even need to be playing at the same time, as in the case of a chess game where only one move might be made per day.

Massively Multiplayer Games on the other hand are usually online role playing games and can involve tens or even hundreds of thousands of people at any given time, leaving players free to join and leave as they please without affecting the progression of others. Prominent examples of these games include Everquest, Ultima Online and The Sims Online, each of which has attracted a loyal following of tens of thousands who are willing to pay monthly subscriptions to play these games for long hours.

Multiplayer architecture

When two or more gamers participate in the same game, it is important that all players see the same representation of the game world and what is happening within it, i.e. they must all agree on the game state. This includes the state of the game terrain, the player controlled characters (PCs), the non player controlled characters (NPCs) and mutable objects such as food, tools and weapons. Without a shared game state, one player may end up trying to attack another player that is no longer present, or pick up an object that has already been moved. In games where the game state changes very rapidly it is imperative that the communication between devices is able to handle the load.

There are two main communication architectures used for multiplayer games.

Client-Server Systems are the most commonly used type of architecture, especially for Massively Multiplayer Games. Centralised servers keep player account information and handle game state; that is players do not exchange directly with each other. The advantages of these systems is that each player needs only maintain a single connection to the server, regardless of the number of players in the game, and each message need only be sent once. This typically means that there is less of a load on network traffic, and servers can further minimise traffic by compressing data before distributing it. Servers also allow centralized control over game conditions and thus prevent cheating and hacking since players are unable to manipulate code that they do not have access to.

However this architecture lacks flexibility and over provisioning is common in order to handle peak loads [11]. Game vendors must have the capacity to handle sudden spikes in load (e.g. on Christmas day when ten thousand people receive their own copy of a particular game and decide to try it out) or risk alienating customers by providing poor service. Although it is unlikely that this peak capacity will be utilized often, it is still needed for this eventuality. If there is insufficient capacity, some players will not be able to play, and others will experience significant lags as the server becomes a bottleneck.

Client server systems can also experience big problems if network latency is high, since all messages sent by clients have to travel a long distance from the client to the server and then from the server to all the other clients.

Peer to Peer Systems promise a lot of advantages over client-server systems. By passing messages directly between players, communication is often faster than client server systems. Since all calculations and communication is handled by the clients themselves, there are no server bottlenecks and the system is able to scale up dynamically with the number of players, avoiding the cost of over provisioning and manually reconfiguring server clusters.

This however means that peer-to-peer games are more vulnerable to hacking and cheating. P2P systems also experience significantly higher traffic levels than do client-server systems. Although the transmission of messages between players is direct, the overall traffic sent increases as a square of the number of players. For example, in a game with eight players, each of those eight players must send messages to seven others, as opposed to the client server case where only one message is sent by each player regardless of the number of players.

How it might work

For this business model, we imagine the scenario where wireless ad hoc devices can be used as a platform to offer wireless multiplayer games with significant benefits over conventional gaming systems.

Games can be segmented in terms of the time required to play each game. Simple single player games for mobile phones such as pinball, tic-tac-toe or snake are very suitable for occupying an empty moment or two whilst waiting for a bus to arrive or a lecture to start.

Multiplayer games by nature require relatively more commitment since they are social activities that are experienced with others. Small Group Multiplayer games require that gamers spend at least the duration of a game on the network, whereas Massive Multiplayer Games usually demand a significant time commitment in order to progress in the game (although not necessarily in one sitting).

Games can also be segmented in terms of their hardware requirements. Simple games which are less hardware intensive can be run on modern mobile phones or PDA's which have the processing power, memory and display capabilities equivalent to that of PC's many years ago. However, multiplayer first person shooters, simulations, strategy games and the like are very demanding in terms of hardware and will require high powered devices, possibly even dedicated games machines, to run well.

For our ad hoc scenarios, three cases are considered. The first scenario is relatively straightforward i.e., *simple games that can be deployed on existing hardware,* the second is *a retail model for complex games that use dedicated hardware,* and finally we have *a rental model for complex games that use dedicated hardware*. Please note that it is possible that all three models be implemented at the same time for different customer segments; however suitable measures would have to be put in place to prevent channel conflict or self-cannibalization.

Model 1: Simple games that can be deployed on existing hardware

In this scenario, games generate marginal revenue in themselves, but drive revenue overall by adding value to existing ad hoc devices, much like games for mobile phones at the moment, with the added advantage of low costs and extensive multiplayer capabilities. It can be envisaged that there might one day be a proliferation of multi purpose ad hoc devices, equipped with instant messaging, organizing, file sharing, voice communication and gaming capabilities. The games on these devices would be relatively simple, and not the sort that customers would pay a significant premium for in themselves.

A user of this device, John, who is on a bus stuck in a traffic jam might thus take out his handy ad hoc device, select a game of his choice and search the ad hoc network for other random players in the vicinity to start a quick multiplayer game with. Alternatively, John might use the instant messaging service to invite a few friends with compatible ad hoc devices to join him.

Once a group of players has been organized, each of the players join the same game and play commences over the ad hoc network. If the user is too far away from certain players, his ad hoc device relays the message via other devices in the vicinity.

If he gets bored of the games that he has on his device, John may choose to temporarily connect to an access point where he can download new games at a marginal cost. The content developers make most of their money from subsidies received from the hardware manufacturers who generate revenue by selling more devices and hence want to add value to their devices so that more customers will buy them.

This is especially important since there is a distinct network effect inherent in these devices. The more people that own the devices, the more useful they become as a whole (more relay points and more people to communicate and share files with). It is important that sufficient promotions and marketing efforts are made to build up a critical mass of devices in the initial stages. Hardware manufacturers are thus happy to subsidise the games in the hope that people who wish to play the games with others will encourage their friends to buy the same devices and generally make the devices more appealing.

The value chain shown in Figure 7.3 is a map of the primary players within Scenario 2, Model 1. These players make contracts between each other to conduct exchanges of information, money, services or a combination of the three. Again, dotted lines indicate the flow of information or services, for example the information that is shared between hardware manufactures and content developers, whereas solid lines indicate a monetary exchange, for example when a customer pays hardware retailers for devices. The value chain map allows the relationship between the major players to be established at a glance and allows a better understanding of where value is or needs to be generated.

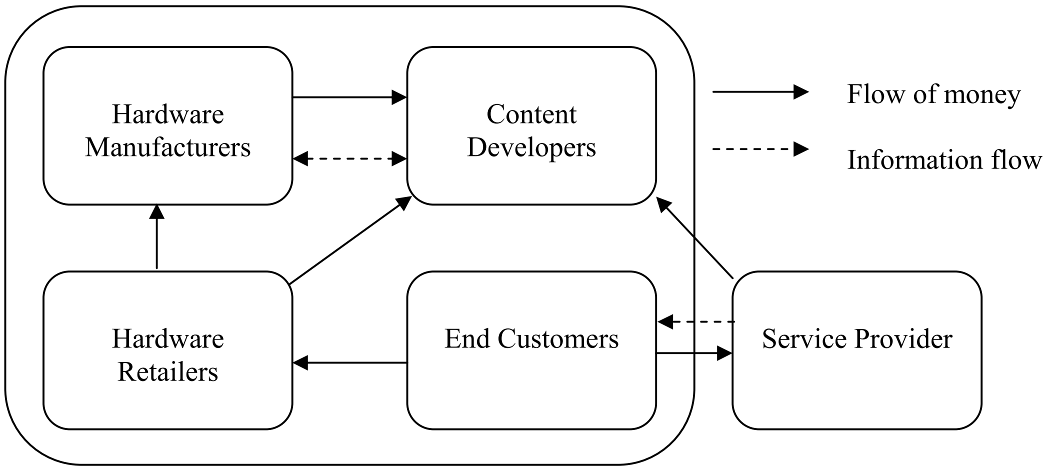


Figure 7.3. The value chain for an ad hoc gaming on ubiquitous devices scenario's primary players

Model 2: Retail model for complex games played on dedicated ad hoc gaming devices

In this scenario, games become the main revenue generator and the ad hoc gaming devices themselves take on a secondary role and might even be sold below cost in order to encourage adoption and thus boost the subsequent purchase of games. This model is similar to the video game console market where manufacturers build a gaming machine with set standards which can then be used as a standardized platform for game developers.

Using this model, hardware manufacturers make most of their revenue through profit sharing or licensing agreements with content developers and publishers. Content developers are responsible for producing the actual games and publishers take on the role of packaging, marketing, distributing and promoting them. Their revenue comes from the sale of the games themselves. Retailers are the main contact point to the customers and add an additional margin.

John therefore goes to his favorite retailer to browse the games section. A particular game catches his eye so he proceeds to the checkout and buys it. On the way home, he loads the game onto his dedicated ad hoc device and is immediately able to connect with other players. He enjoys the game so much that he decides to continue playing after he reaches home.

John is not alone in wanting to play wireless games at home. Even with current mobile devices, according to In-fuso analysts [19], the average session time for a mobile gamer is 22 minutes, and 40% play for over an hour. In other words, mobile games are not just for people waiting at bus stops. Indeed, 75% of mobile gamers are playing at home and during the weekends.

In order to facilitate these habits, and to allow better enjoyment of complex games, dedicated ad hoc gaming devices will be designed with docking capabilities. John is thus able to plug his device into the television and sound system at home, as well as having the choice of adding additional control devices such as keyboards, steering wheels, mice, and joysticks with force feedback.

This will allow Massive Multiplayer Online Role Playing games, which would be impractical to play on the move, to be played through the ad hoc network. Through playing his favourite games at home, John has been able to establish a network of gaming friends who all live in the same neighborhood.

The value chain shown in Figure 7.4 is a map of the primary players within Scenario 2, Model 2.

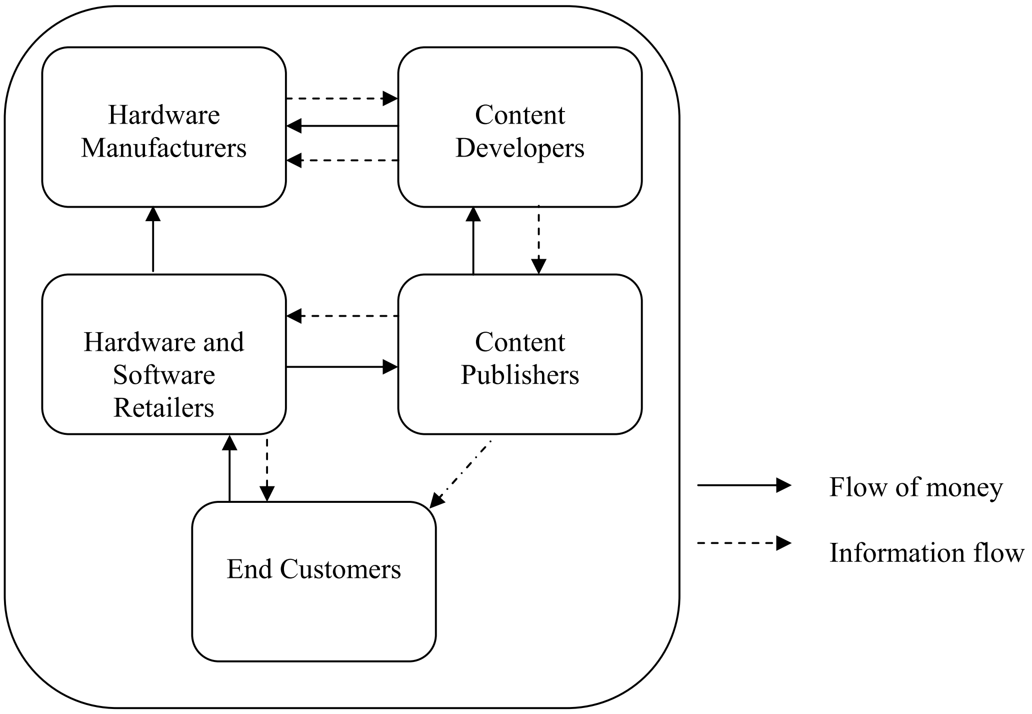


Figure 7.4. The value chain for a retail based dedicated ad hoc gaming scenario's primary players

Model 3: Rental model for complex games played on dedicated ad hoc gaming devices

This scenario is the most unconventional model of the three, but potentially the one that generates the most recurring value all around. The model can also be used in parallel with the other two. In this model, consumers do not buy the ad hoc gaming devices, nor do they buy the games, rather, they rent them as one.

From the success of video game rentals and subscription based Massive Multiplayer Games, it seems likely that consumers would be receptive to this model of gaming.
The upfront cost to consumers is much less, nor will they need to pay for any features or games that they do not truly want. Hardware manufacturers and content providers on the other hand will be able to work together to provide dedicated devices with specific games loaded onto them that can be rented out for long or short periods depending on the game, thus generating significant recurring income, spreading costs over a greater number of customers and retaining control over all aspects of the system. Necessary hardware and software upgrades can be done automatically without any cooperation from end users. This could therefore prove to be a better long term model than the previous scenarios where income is only generated once and players are able to play over the ad hoc network for free thereafter.

John is now able to go to his neighborhood video rental store with two friends and rent the latest first person shooter game. At the counter, the clerk brings out three ad hoc gaming devices preloaded with the game in question and checks them out for John and his friends. One of John's friends then spots another game and expresses interest in it, and they decide to rent that game as well. The clerk is able to quickly load the additional game onto the devices and charge them accordingly for it.

Upon leaving the store they are immediately able to start playing with each other and any other players with the same games. As with the previous scenario they are able to go home and plug the devices into their televisions and other devices for a more engaging experience.

The value chain shown in Figure 7.5 is a map of the primary players within Scenario 2, Model 3.

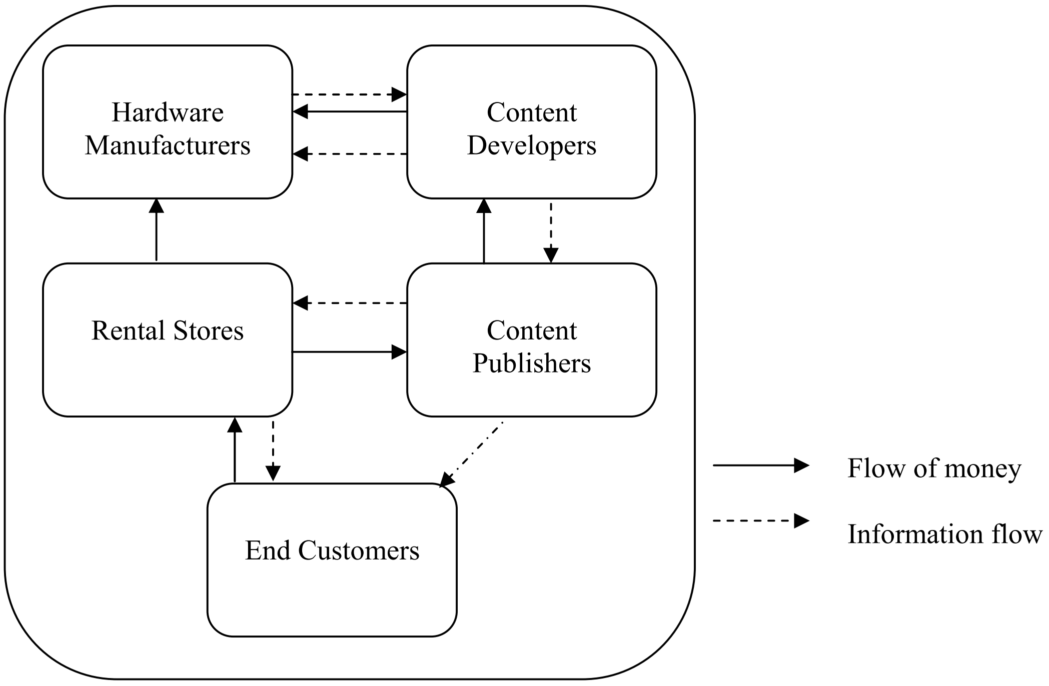


Figure 7.5. The value chain for a retail based dedicated ad hoc gaming scenario's primary players

Value proposition for primary players involved

For customers:

Model 1:

At marginal or no cost to them, users are able to enjoy enhanced value from their ad hoc devices. Even during brief moments of downtime such as during commutes, users are able to enjoy social interaction through games.
Makes it easy to contact new or existing friends to link up and share a game with zero operating costs.

Model 2:

Consumers who have no broadband connections and who cannot afford steep carrier charges can now enjoy multiplayer games at a relatively low cost.

With dedicated hardware, users are able to experience the merged convenience of wireless ad hoc devices together with a powerful games machine that can function on the move as well as at home.
Subsidised hardware means that users pay less up front.
After purchasing a particular game up front, there are no further subscription costs or service provider charges to pay. The average cost of playing a game thus decreases proportionally with usage and is well suited to gamers who enjoy spending long hours on the same game.

Model 3:

The rental model means that users pay no up front costs, merely the rental price of the games themselves.
Consumers who have no broadband connections and who cannot afford steep carrier charges can now enjoy multiplayer games at a relatively low cost.
By renting devices, the customer is always ensured of having the latest and most powerful hardware available.
A range of pricing tariffs enables gamers to rent according to the way that they play, e.g. higher rentals for arcade games which are only played for a few days of one-off fun and lower rentals for longer term games such as role playing games.
The relatively low one off costs and ability to rent for short periods allows gamers more flexibility and the opportunity to play many more games than they might try in the retail model.

For hardware manufacturers:

Model 1:

By adding better multiplayer games onto their devices, hardware manufacturers enhance the inherent value of their devices at a marginal cost. The enhanced functionality of the product makes it more appealing and may thus boost sales both directly and indirectly.

Model 2:

Provided they successfully sell enough units of their ad hoc gaming devices, there is considerable upside to be gained from profit sharing and licensing agreements with content providers, publishers and retailers. Once critical mass is achieved and a large number of games are being sold there will be a steady stream of revenue coming in at a higher profit margin than could have been achieved by selling the hardware alone at a premium.
Licensing deals hold the potential of deals involving spin off merchandise or products based on popular games.

Model 3:

Since hardware manufacturers need to subsidise their hardware in order to make their money from content, it makes even more sense for them to rent their hardware complete with content and thus spread out the cost of each unit over a greater number of customers.
Since the rental model will tend to be lucrative for good games (consumers will play them for longer and more consumers will want them), developers will earn more revenue if their games are good. Overall, this should provide the incentives for all developers to improve the quality of their games, all of which will increase the value of the hardware manufacturer’s devices and hence increase their sales volume, as well as the subsequent share in developer’s profits.

For content developers:

All models:

With an additional platform to write for, content developers are able to spread the cost of development over a greater target audience (withstanding exclusivity agreements)

Models 2 & 3:

The convenience of a standardized ad hoc gaming device means that developers can write efficient games that fully utilize a gaming machine's power rather than worry about differing incompatible standards of graphics processors, sound cards, processors, operating systems and so on. Communications between standardized devices will also be simplified, negating the need to write for a variety of incompatible protocols.
Writing for a truly wireless networked device will allow developers to come up with a new range of games exploiting mobility and connectivity. With location finding capability this model can be extended even more. New generations of location based games are already being developed to take advantage of location finding capabilities. [12]

Model 3:

The rental based model may bring more returns to the developer in the long run, as revenue is recurring, especially for games which require a long time to complete.
The cost of developing a multiplayer game is high but the cost of producing each subsequent copy of the game is very low. It therefore makes sense to maximize the number of players who play to achieve critical mass (the more people that play the game, the more value the game has to each player). The low cost of renting a game may thus encourage more people to play.
Piracy becomes virtually nonexistent with this model as it is the devices that are rented, not the games. It thus makes the games impossible to copy as consumers will not have devices to copy the games on to. In any case, the nature of multiplayer games makes it easier to institute checks to ensure the software authenticity of participating users.
With a rental model, there is far more incentive to improve the quality of games so that more consumers will rent them for longer periods of time. An analogy can be made between a blockbuster movie that is released with great fanfare only to disappoint fans and lose subsequent revenue through bad word of mouth, as opposed to a movie that is released on video but through word of mouth becomes a cult classic, generating huge revenues. This gives developers more power over publishers who may have conflicting agendas, and incentives to compromise the quality of releases in order to meet deadlines and reduce costs.

Risks & Disadvantages

Competitors:

In the first scenario, ad hoc communications systems are an enabling technology that can potentially improve the way cab companies communicate. In the multiplayer gaming scenario however, ad hoc devices completely change the business model for the players involved.

The biggest difference between these models and conventional mobile service provider models is that the dominant link, the carrier, has been removed, which dramatically changes the value chain. Where the carrier was responsible for marketing, distribution and billing of services, these functions are now shared amongst other players, resulting in more value throughout the value chain.

Whilst this brings many benefits as detailed in previous sections, the biggest drawback to this is that conventional carriers will necessarily see this service as a threatening competitor and are likely to react aggressively to protect their market share. This is doubly true in the case of mobile providers that have expended huge resources in building 3G networks for the same purpose that the ad hoc networks in our model are attempting to provide at a far lower cost. With such a large investment to protect, they may be forced to act unduly aggressively. It would be unwise to provoke a price war since it would benefit no one other than the consumer in the end.

Existing video game console manufacturers will also be threatened by ad hoc gaming devices which compete directly with their product offerings, as well as for the attention of content developers. The biggest players will have secured as many exclusive deals with the top content developers as possible, which could lead to new entrants being locked out, unless they have sufficient clout and deep pockets. Small developers on the other hand may not wish to risk their resources developing on a new, unproven system.

It is therefore of vital importance that sufficient efforts and incentives be made to entice all players involved to work with each other in all aspects including hardware design, content development, marketing, promotion and distribution by forming partnerships and/or a consortium. In an industry where content is king, it is vital that an impressive stable of developers be gathered to produce games for the system. It is also likely that only with the leverage of a combined consortium will a new entrant into the gaming industry be able to withstand the intense competition from entrenched players, as well as from conventional mobile operators.

Coverage and adoption issues:

By the nature of being an ad hoc system, good coverage depends on having a suitably spread out number of nodes around the intended communications area. This means that to work properly, there must first of all be a sufficient number of devices to cover the entire service area.

This makes the network effect doubly potent for ad hoc gaming devices. Not only is it true that the more players play a multiplayer game the more interesting the game is, but for the ad hoc case, without sufficient players (or at least functioning devices) to act as relay points, no multiplayer gaming will be possible at all in some locations.

This means that until the devices become truly ubiquitous, they will at best only be able to perform well within city or town limits where there is a high density of people and hence devices. Taken into low population density areas, or areas where the devices are unpopular, the gaming devices would be left virtually useless as multiplayer devices. It is therefore important that a large scale effort be made in each location the device is launched to push as many devices as possible to local consumers. The rental model is particularly suitable here, since customers will generally tend to be from the neighborhood in any case, decreasing the likelihood that too many users will venture out of the coverage area at the same time. The devices stored in rental stores (located at strategic positions) would themselves be able to act as relay points. For those cases where broken coverage is inevitable, good single player games/modes must also be provided, e.g. for camping trips where only one device is present.

In addition to cases where there is no coverage, it is important to ensure as far as possible that despite the unpredictable QoS inherent in ad-hoc devices, an overall acceptable level of connection can be maintained for players already involved in a game. If games are frequently disrupted or experience lagging due to bad QoS, customers will switch to competitors, especially in the rental model as switching costs are very low.

The system also needs to be fairly robust, so that one player who is experiencing latency problems does not slow down the entire game. Techniques such as dead reckoning may help reduce the feeling of lag overall.

Finally, by nature ad hoc gaming devices will be much more localized than gaming over the Internet where you might as easily be playing with someone from India as from China. However, as the purpose is social gaming as opposed to communicating, this may not be a severe drawback and in fact makes it more likely that players will share a common sense of gaming etiquette and want to play at similar times of the day. It may also foster the formation of real friendships that can be strengthened through face to face meetings which would be more difficult in long distance cases.

Security issues:

Despite the benefits that P2P architectures offer, as explained in previous sections, they are not as secure as client-server systems and are vulnerable to hacking and cheating which can decrease the level of enjoyment for other players. The problem can be reduced by attempting to make the designs more secure, or by instituting controlling mechanisms where participating peers validate each other’s behaviour, but it is unlikely that the problem can be completely overcome easily.

Scalability issues:

The entire system must be scalable to avoid potential problems in the future. With limited bandwidth in a small area, there is always a limit to the number of users that can be supported, and therefore should the devices become truly successful, it may inherently cause problems with the effectiveness of the system.

Massive Multiplayer Games pose a particular problem since for peer to peer games, communication demands scale as a function of the number of players squared. If a game should have a few thousand players involved, the sheer volume of traffic could cripple the entire ad hoc network, including the devices of players not within the game who are acting as relay points. While software algorithms might be able to significantly reduce the amount of traffic by intelligently selecting only necessary information to transmit, there is still a theoretical limit to the number of players that can be supported in any given game.

Therefore, if necessary, Client Server configurations might be needed for certain games. The communications demands in this case scale linearly and if more capacity is needed additional servers can be added. However, this brings its own problems as sufficient servers would need to be located around the city to ensure coverage.

Cost issues:

For the retail model, subsidizing ad hoc devices may be a necessary step to achieving critical mass, but runs the risk of losing more money than revenue from licensing and profit sharing can cover. It is also virtually impossible to price low in the beginning to build volume and later raise the price because consumers of electronics goods are accustomed to only seeing price drops. Pricing pressure from competitors will also force prices down.

For the rental model, pricing strategies must be analyzed very carefully so as to be low enough to encourage large volumes of rentals, yet high enough to make a profit over a reasonable rental lifetime of each device. Costs will comprise not only that of the devices themselves, but also the cost of maintaining and repairing damaged units. As these devices will be quite valuable, some customers may not wish to return their devices. Mechanisms such as collecting credit card details must be in place to discourage this sort of damage or failure to return devices, without unduly inconveniencing customers.

In light of the risks detailed previously in this section, a detailed cost benefit analysis should be done to ensure that the long-term benefits outweigh all aspects of potential risk and cost involved.

Incentives issue:

In the first model with multi purpose devices, most users will tend to keep their devices on in order to receive email or instant messages. However, for the models using dedicated ad hoc gaming devices, there is little reason for an individual user to allow his device to be used as a relay if he is not playing a game at that moment in time. When the user is not playing a game on a conventional video game console he might well shut it off rather than waste power on it. However this behaviour needs to be changed for the ad hoc model to work which is why designing incentive systems become extremely important here. Without sufficient incentives for individual users to keep the majority of devices within the service area operating as relay points, network coverage would be severely compromised.

Shopping Mall Scenario

The days where it was feared that online shopping would completely replace brick and mortar stores are long past. Shopping is an undeniably social experience that is as much about being with friends and family and enjoying the atmosphere as it is about purchasing items. Shopping at stores also allows consumers to try things out before buying them which is important with items such as clothing and footwear. The ability to browse and induce impulse buying is also an important factor in retail that is not immediately translatable to the online space where banner or pop up ads have to be used to draw attention rather than allowing consumers to simply come across interesting items. Finally, some items are simply not economical to keep in inventory and to deliver to consumers as is required in the online retail model. However, at the same time there is much to be said for the convenience and ease that online shopping brings to consumers.

Shopping malls provide one stop shopping solutions to consumers by aggregating a large number of retailers under one roof in a pleasant environment. The worldwide trend is towards bigger and better shopping malls with hundreds of shops spanning multiple floors over a large area. While this large selection is beneficial to consumers, in practical terms it causes some inconveniences as consumers may have trouble getting around and finding the items and shops that they want amongst the hundreds that are available. The retailers on the other hand may face trouble drawing traffic to their shops, especially if they are located in remote areas of the shopping mall and thus suffer from lost opportunity costs when potential customers don’t purchase from them simply because they cannot locate them, or are unaware that they stock particular items.

The solutions that most shopping malls provide are Information Counters where staff point consumers in the right direction and Store Maps that are located at various points around the mall.

While these measures help somewhat, they are usually inefficient as the limited number of staff members are unable to cope with large numbers of queries at any given time. They are also usually unable to give advice on which stores may carry certain items beyond the broad categories listed on the maps and will certainly be unable to inform consumers about whether the desired items are in stock or not. Verbal directions given may also be vague, resulting in frustration for consumers.

How it might work:

In this scenario we envisage merging the advantages of shopping malls with the convenience and information capabilities of shopping online. Our user John, his wife Jane, and their two teenage sons, Mark and David decide to go Christmas shopping at the newly opened TechnoMall. Upon entering the TechnoMall, all four family members are presented with an ad hoc device. With a few button presses, they mate their devices so that they are recognized as a group and can communicate directly. (an alternative to this scenario is the ubiquitous scenario where ad hoc devices are common everyday items that consumers would already have, in which case the devices would be able to hook into the shopping mall's network automatically as soon as the consumers entered the mall).

Immediately, the two boys decide to go off on their own to look at the latest video game devices, while Jane decides to go shopping for clothes. Watching his family disappearing off into the crowd, John wants to reminds them that they are meeting for dinner in a couple of hours but they are already gone.

At any rate, John has some shopping of his own to do, specifically presents for his family. As Jane has helpfully left the same newspaper advert on his desk for the past two weeks, John already knows what he is buying for her, but not being a regular shopper he has no idea where to get it from. Using the ad hoc device which greets him with colorful messages and adverts, he enters details of the product that he wants and is immediately shown a list of shops within the mall that carry it, as well as consumer ratings for the product and the shops. Another tap and he is shown the shop that is nearest to his current location and a map that shows him exactly how to get there.

Minutes later he has purchased his wife's gift and is heading back into the mall as a beep on his device signals an arriving chat request from Jane. He accepts the chat and discovers that Jane has found a shirt that he thinks he would like. Using the product link provided by her, John connects to the relevant shop's ad hoc site and calls up an image of the shirt in question. Finding it to his taste he gives his wife the go ahead and logs out of the chat.

Remembering that he has run out of his favourite work time snacks, he heads down to the supermarket that is housed within the mall. Faced with aisle upon aisle of goods, he again turns to the ad hoc device which immediately shows him the location of the snack in question, informs him of the latest special offers within the supermarket and where the express checkout counters are located.

Having purchased his snacks, John now has a problem; he has no idea what he should buy for his sons. Using the ad hoc device he decides to get help from other shoppers. John is not alone, and there are plenty of pages and chat rooms dedicated to gift suggestions. He is even able to access a real time updated list of the most popular products being purchased within the mall in various categories. After a few minutes he is able to gather that the latest craze for teenage boys is ad hoc gaming, and he decides to buy a pair of devices for them. Using his device he is shown the shops that carry the gaming devices. This time however he does not head to the nearest shop as it has a bad consumer rating due to its poor selection of games.

Following the interactive map, he walks through the mall enjoying the atmosphere until a beep from his ad hoc device suddenly alerts him that he is passing a bookshop that is having a special one hour sale. Pleased to have stumbled upon the sale, John enters the bookshop and comes out with a huge bagful of books. He then continues on his way to the games shop, on the way receiving a few more location based adverts as well as some specific ads based on his purchases to date and that of his family's. However, none of these interest him and he simply ignores them and buys the ad hoc gaming devices for his sons. (The shopping mall's ad hoc devices would be proprietary and not compatible with other devices)

It is now getting late and John is hungry. He sends a message to his entire family asking them what they would like to eat. Jane replies back immediately that she wants Japanese food, however, there is no response from his sons. John rolls his eyes, and presses a few keys on his ad hoc device which locates his sons and points out their location on the interactive map. They are in the arcade, which is probably why they can't hear the beeps from their ad hoc devices. He sends the map of the arcade to his wife and tells her that he will meet her there to collect their sons. As he walks, John searches for a list of Japanese restaurants within the mall and quickly finds one that is highly rated by consumers. He initiates a brief messaging conversation with the Japanese restaurant to make a booking and finds out that the special today is grilled fish.

An hour and a half later, John's family is fed and happy, having enjoyed a fun day at the TechnoMall. At the exit they return their ad hoc devices, which are quickly cleaned, reset, charged, and ready for use by the next set of visitors.

The value chain shown in Figure 7.6 is a map of the primary players within the shopping mall scenario. These players make contracts between each other to conduct exchanges of information, money, services or a combination of the three. Dotted lines indicate the flow of information or services, for example the information that is shared between hardware and software suppliers, whereas solid lines indicate a monetary exchange, for example when the shopping mall pays hardware suppliers for devices. The value chain map allows the relationship between the major players to be established at a glance and allows a better understanding of where value is or needs to be generated.

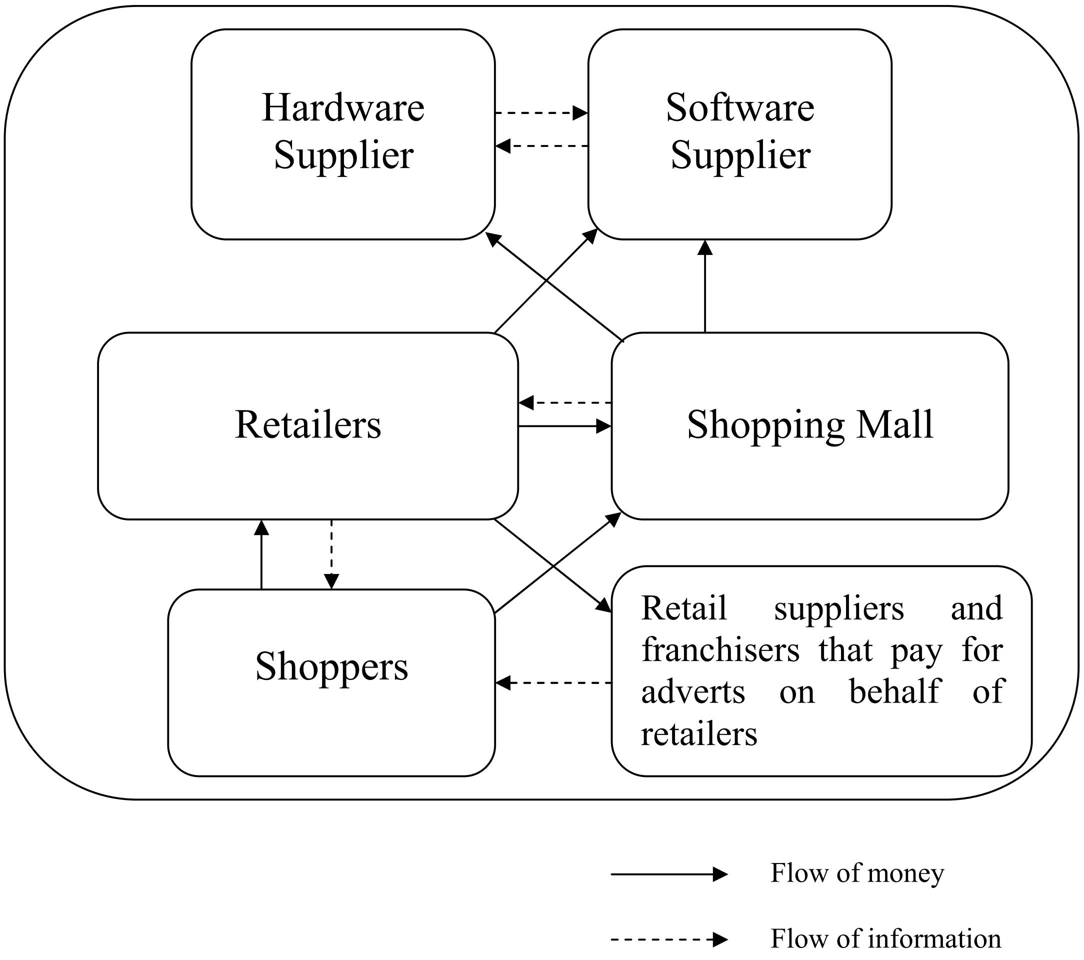


Figure 7.6. The value chain for an ad hoc shopping mall network scenario's primary players

Value proposition for primary players involved

For shoppers:

Shoppers are able to communicate not only with friends, but also with strangers within the mall that may share common interests with them through the use of instant messaging, topical chat rooms and message boards. They are also able to easily make inquiries with shops before walking there to make bookings or ask about stock etc.
Shoppers are able to share useful information with present and future shoppers by rating shops and products and leaving comments on relevant message boards. This not only means that shoppers have a much better idea of which products are good and where to go within the shopping mall, but also increases the power that consumers have over retailers. Consumers are now able to reward or penalize shops or products through optional rating systems which gives retailers the incentive to stock better products and provide better service. With information freely available through ad hoc devices, a shop which consistently receives bad ratings and does nothing to improve will soon see its revenue take a significant hit.

By using the product search function as well as the interactive maps, shoppers are able to instantly locate the products they desire and obtain a list of shops that carry them. This can be extended to a greater level of detail for larger shops within the mall. For example, a shopper in a supermarket is able to locate desired items to the nearest aisle, allowing shoppers in a hurry to head straight for what they want and leave.
With location finding capabilities, ad hoc devices allow people to find each other easily within large crowded shopping malls, and are especially useful in basement levels where there is often no reception for mobile phones.
Location specific advertising can also be a useful function that alerts consumers to particular offers or services in their vicinity.

For shopping mall owners:

By providing services to shoppers that increase the convenience and enjoyment of their shopping experience at no additional cost to shoppers, the value of their shopping mall is increased since it should attract more shoppers who will spend more money. This results in an increase in both direct revenue (e.g. car parking fees, mall owned restaurant revenues) and indirect revenue (the ability to charge higher rental from retailers based on high traffic).
By increasing the speed and efficiency with which shoppers shop, the shopping mall is able to increase turnover of shoppers by allowing shoppers who only want a couple of specific items to quickly finish their business and leave, freeing up space for new shoppers.
A number of additional revenue streams can be generated through the use of ad hoc devices, the first of which is charging retailers for the provision of the ad hoc device service. The amount charged per retailer depends on the amount of utility each retailer gains from being connected to the network and the services that they subscribe to. For example, shops might be charged a certain amount per customer that is routed to it through queries on the ad hoc device and interactive maps. Retailers such as supermarkets who have detailed aisle listing and a greater selection of products to be stored would also incur higher costs. Ideally the individual amounts involved would be small, but would collectively mount up to be a significant source of revenue.
Another additional revenue stream is generated through the charging of ads that appear on the ad hoc device. Ads will again be priced according to the utility gained by the retailers. Some ads may be paid for by product suppliers or franchisers on behalf of the retailer or franchise operator.
Finally, a great deal of marketing information can be gained through monitoring the ad hoc devices. This could include information about which products and shops are queried, the shopping habits of individuals, and even how shoppers move about in the mall, through the use of location tracking. Due to privacy issues, none of the information collected would be able to be linked to any particular individual; in fact the devices are handed out to incoming shoppers randomly, and there is no record of which shopper takes which device. This anonymous but highly accurate marketing information could then be sold to retailers within the mall, or third parties outside the mall.

For retailers:

Through the increase of overall traffic to the shopping mall, individual retailers should see an overall increase in traffic to their shops. More importantly, through the use of the ad hoc query and mapping system, more relevant customers will be routed to their shops, as opposed to mere browsers, resulting in a higher customer conversion rate and thus higher revenues.
The convenient mapping system makes it less important for retailers to pay higher rents for prime positions in the mall, as customers will easily be able to be routed to more remote areas.
Retailers are able to respond to customer queries quickly through the messaging system and work with customers even before they are in the shop.

Advertising shown to people who are actually shopping is likely to result in more conversions than higher cost blanket advertising used in other forms of media. Location specific adverts may prove even more effective, especially when used in conjunction with short term promotions.
Marketing information collected through the ad hoc network, combined with consumer rating systems and message boards allow retailers to closely monitor the pulse of consumers and adapt their services and product offerings accordingly to what their customers want, hence improving their reputation and earnings.

For hardware suppliers:

Hardware suppliers are able to achieve large bulk orders through the sale of devices to shopping malls, which can be repeated every time technology has progressed significantly enough to warrant an upgrade.
Regular revenue is generated through the repair and refurbishing of existing units.

Risks and Disadvantages

Coverage and scalability issues:

In the shopping mall scenario, coverage issues are much less severe than previous scenarios since the coverage area is confined within the boundaries of the mall. Even in the early morning or late night when only a few shoppers are about, there is little danger that there will not be enough relay nodes, considering the fact that all shops within the mall will be connected to the ad hoc network.

However, the flipside of the problem is that it is much more likely in this scenario that a great number of ad hoc devices could congregate in a small area resulting in interference. If a popular event were to be held, such as a performance in the centre of the shopping mall, interference could cripple all the devices in that area and hence hamper the communication efforts of other users as well who need to relay through that point. This could also mean that there is an upper limit to how much the system could scale within a particular mall. Once a certain number of devices have been checked out, subsequent shoppers may have to enter the mall without devices to minimise the probability of interference occurring.

Costs issue:

At any given time a large shopping mall may have well over ten thousand shoppers in it. Providing each and every one of them with a device, as well as keeping enough extra units to absorb occasional peak loads and replacements for broken units will be a considerable cost.

There are also maintenance costs, repair costs for broken units, refurbishment costs and extra staff required to check the devices in and out at the entrances and exits.

However, with economies of scale and suitable pricing schemes, it is likely that the revenue generated from these devices will outweigh the costs involved. A detailed cost benefit analysis should be performed to ensure that the long-term benefits outweigh the potential costs and risks involved.

Consumer power issue:

Ideally, shoppers would be able to use the information provided to interact socially and help each other to enhance their shopping experiences. However, inevitably, there would be abuses to the system such as unwanted messaging harassment and the posting of obscene messages.

It can also be imagined that shoppers might choose to take the consumer ratings system one step further and post too much information on message boards. For example it could give shoppers the ability to list the different prices that every store charged for the same product, allowing consumers the ability to comparison shop instantly. Transparency of information in this case could lead to severe problems with retailers as this would force them to gravitate towards the lowest price and compete by undercutting each other, resulting in a price war that would benefit no one but the consumers.

These problems would have to be avoided through the use of moderators and other enforcement mechanisms.

Adoption issues:

A necessary step to implementing this business model is to achieve buy in from all the players involved. The system is only useful if most if not all retailers and consumers use it, as the available information increases proportionally. Retailers are especially important as without information about the entire contents of the shopping mall the system would be limited and far less useful, meaning that consumers would be far less likely to use them. Some retailers might well refuse to join the system for various reasons. These reasons might include unwillingness to pay for inclusion in the system for those that feel they would not gain significant benefit from it, or those who fear that the information provided by the system would undermine their competitiveness, or simply because they don’t believe in the system.

To pre-empt these problems, all occupants of the shopping mall would need to be educated about the benefits of the system and have their concerns addressed. The pricing system for inclusion needs to be carefully designed so that the amount paid is in line with the amount of benefit or utility gained. For newer shopping malls especially, it may be possible to include subscription to the service in the rental terms. Once the retailers have signed on, it will then be necessary to ensure that sufficient high quality content is available, and that customers are educated on how to access that content and make use of the ad hoc devices.

Substitutes and alternatives:

Alternative technologies to wireless ad hoc devices could threaten the feasibility of this business model if the alternatives were able to perform at an equal or better level and at an equal or lower cost. For example, if wireless LAN’s could be made to hand off seamlessly and enough base stations were situated throughout the mall, it is possible that the theoretical quality of service might be better.

However, in practical terms, given the limited area of a shopping mall and the sheer number of redundant relay points, ad hoc devices would have a high chance of performing well, perhaps even more so than wireless LAN’s which would experience localized outages if one or more of the basestations were to go offline. Similarly, the low maintenance and self-organizing qualities of ad hoc networks would make them cheaper to run than a centralized system.

Shrinkage and damage issues:

By necessity of function, the mall ad hoc devices would be relatively expensive and functional devices and not something that would usually be handed out to each and every customer, albeit on a temporary basis. The risk is that a significant number of shoppers would feel compelled to steal the devices or damage them in some way, either by accident or on purpose.

The risk of damage is not something that can easily be prevented other than by ensuring the design is sufficiently robust. At the very least the devices should be able to handle drops of significant force that are sure to happen. A penalty can also be imposed on shoppers who return devices that are obviously in much worse shape than they were when checked out.

The risk of theft can be mitigated somewhat by designing the device such that it is completely useless outside of the shopping mall. However, a savvy thief might still be compelled to steal the devices for its parts rather than its whole. Security tags should therefore be embedded in the devices to alert personnel should the devices be taken away. Location enabled devices will be able to signal themselves should they be brought outside mall boundaries.

Finally, it is possible that a deposit system be instituted. However, this has drawbacks in that it would certainly result in bottlenecks at the entrance and exit to the malls as devices are checked in and out and deposits are collected and returned. Many shoppers would feel inclined to skip this process and proceed without the ad hoc devices thus making the system as a whole pointless. Also, the deposit would have to be an amount small enough so as not to deter shoppers yet large enough that they would rather get it back than steal a device that would be useless to them.

None of these risks would be a problem in the alternative ubiquitous scenario where each shopper would be bringing their own device.

7.2. Conclusions & Key Recommendations

In the previous sections we have evaluated the strengths and weaknesses of MANET's, segmented the potential types of ad hoc network and identified a number of scenarios where MANET's might be suitable. Through the analysis of these scenarios, we can draw a number of conclusions about where and how MANET's should be used in order to achieve financially viable networks.

One of the most important things to note is that ad hoc networks are drastically different from conventional carrier networks in a business sense. While carrier networks are a business in their own right, and the service of providing connectivity can be sold on its own, ad hoc networks act more as an enabling technology and connectivity is by nature free. Ad hoc networks therefore bring value to players involved in a different way to carrier networks, by enabling lower cost, lower maintenance networks to be deployed in areas where centralized infrastructures would be impossible, undesirable or unnecessary. It can be seen through the very different scenarios presented that revenue is not generated from the provision of the service itself, but rather through the enhanced value that it brings to the application it is used for.

This is an important consideration to keep in mind when deciding where to implement ad hoc networks. In circumstances such as emergency services ad hoc networks can and should be purely altruistic, but in commercial circumstances ad hoc networks must be able to generate value for players involved and justify the cost of implementation as well as its benefits over alternative technologies.

In most cases other than basic communication applications between a relatively small number of people, some form of centralization is necessary for practical business models. This does not mean that content should always be centralized as that detracts from the power of ad hoc networks, but rather that some form of centralized control is necessary in order to have a viable business model. In the shopping mall scenario for example, the devices were kept central and thus under control of the shopping mall management whereas the content was kept decentralized throughout the shops and the consumers themselves. In the gaming scenarios, either the games or the devices themselves were kept centralized. This method allows a central body to absorb the costs of implementation involved, by charging revenue for the services that it provides. Without some form of centralization it is very difficult to achieve a financially viable network as the entire service is free to use and there are no real incentives for any player to bear the costs of providing content or hardware beyond basic devices.

Regulation has an important part to play in ad hoc networks. Bandwidth needs to be allocated and regulated to promote the growth and use of ad hoc networks while at the same time making sure that the networks are able to maintain a reasonable level of QoS and do not cause harmful interference to others.

Since these devices may potentially be used to transmit and receive information of varying sensitivity, ranging from private messages sent over instant messaging services to credit card information, the security of communication needs to be seriously addressed in order for ad hoc devices to have any reasonable commercial applications outside of very basic services. Even gaming services need to be protected sufficiently to prevent hacking and cheating activities that might undermine the overall integrity of the network and drive other consumers away. For applications where ad hoc devices are rented or lent temporarily to consumers, mechanisms need to be designed to prevent theft or breakage.

Another key challenge for ad hoc technology in any business application is to encourage a large enough number of early adopters to use the technology in order to build up critical mass. Without enough people using the technology from the outset, an ad hoc network will have very limited uses, and in many cases will not work at all as there will not be enough nodes to relay messages with. This means that applications where a large initial number of users can be harnessed immediately such as the shopping mall scenario are more promising as pilot projects, whereas applications such as those described in the gaming scenarios may need to be phased in at a slower rate to avoid a multiplayer gaming network where there is no one else to play with. In any case, for consumer applications, media coverage, large scale launches, discounts and promotions will all be necessary to encourage adoption. Individual user applications will need to be included for many consumer applications to encourage adoption especially in early stages where a comprehensive network may not yet be available.

Incentive systems also have a very important part in the success of ad hoc networks. Rather uniquely, MANET's are networks that are built on cooperation. Without a sufficient number of nodes cooperatively sharing resources, an ad hoc network cannot be made to work. If left alone, most nodes would probably try to maximize their own utility by not sharing their power and bandwidth resources which would subsequently cause problems with the network. Incentive systems must thereby be designed to encourage users not only to use the devices but also to allow other users to relay off of them for their own good as well as the greater good.

All of the above factors need to be taken into consideration as the MobileMAN project proceeds in order to bring the technology from theory to practice in a feasible manner. However, with sufficient resources devoted to these issues, ad hoc devices appear to offer some compelling benefits to all sorts of users.

7.3. Cooperation Enforcing: an Economic Perspective

In this section we analyze the problem of cooperation within mobile ad hoc networks from an economic perspective. This analysis is performed by observing this problem with reference to the adoption cycle for mobile ad hoc networks. From this analysis we conclude that there is no an economic value for pushing the users to not cooperate, and hence we probably will not need incentive systems anyway

In order for mobile ad hoc networks or indeed any new technology to move from concept to reality, it needs to go through successive phases of development, deployment and adoption in order to eventually achieve critical mass and enter the mainstream market. At each phase of technology adoption, there is a different target customer segment with different needs and preferences. Solutions should therefore be designed and implemented with each segment's unique needs in mind. For ad hoc networks in particular, there is a need to work in distinct phases with the aim of steadily building up users. There is a chicken and egg situation where the usefulness of the network increases with the number of users forming and contributing to the network, but without enough users joining in initially, it will not be useful enough to attract more users. That is why a phased deployment makes much more sense than a full-scale deployment out of hand. Trying to run before being able to walk may result in the technology never taking off at all.
Unfortunately, current research into mobile ad hoc networks have mostly been directed with the assumptions that the networks will be mainly used for large scale general consumer applications, and that nodes will be ubiquitous and reasonably dense. Both of these assumptions are considerably far from reality and will certainly not be true for initial phases of deployment; if the networks are designed and implemented with these assumptions in mind they run a high risk of failing. It is unreasonable to make plans for a bright future without first considering how to get there in the first place; the needs of the early market must not be ignored.
Given the strengths and weaknesses of ad hoc networks, it is unlikely that they will be able to be deployed on a large scale for general applications until much further down the adoption cycle. In the early stages, it is much more reasonable to expect ad hoc networks to be used for specific applications which fully capitalize on their strengths, with solutions that are both useful and financially sustainable[18]. In the same vein, it is unrealistic to expect a sudden proliferation of devices with networks of hundreds or thousands of nodes, especially with general applications that do not belong to a single authority.
In order to bootstrap adoption of the technology, it is therefore imperative that issues such as overly complex incentive systems do not cause early adopters of the technology to shun it. Early stage networks will most likely either be formed for specific applications under a single authority, where incentives are not needed, or by small groups of pioneering, technologically savvy users.
We therefore envisage a solution that evolves according to the adoption cycle of mobile ad hoc networks, loosely based on Geoffrey Moore's Crossing the Chasm model [19]. In the earliest stage, we expect users to mainly be comprised of *pioneers*, technologically savvy users who are very enthusiastic about new technology and are more interested in exploring technology than actually benefiting from it. These users are very cooperative by nature and in addition are likely to be much more forgiving of faults in developing technologies; in many cases actually contributing to its development. We can draw parallels with the case of Peer-to-Peer networks, which usually see an extremely high level of cooperation in early days, and degrade slowly as they become more mainstream and attract more general users.
At this stage, we argue that incentive systems are not needed at all; the desired behaviour for nodes can simply be hardwired into nodes at hardware as well as a protocol level and trust that

the majority of users will not tamper with the devices. This will avoid all the problems discussed previously, ease implementation, reduce complexity and allow all forwarding functions to be handled automatically within the network for it to be fully self-organizing.

Pioneering users have little incentive to hack the system and early applications are likely to be both specific and limited to small groups of users with common goals. By reducing problems and limitations for users, pioneers will become champions of the technology and introduce it to the next customer segment down the adoption cycle, the *visionaries*.

Visionaries are different from pioneers in that they are not interested in technology for technology's sake but rather see the potential in new technology and are willing to make sacrifices in order to be amongst the first to see that potential realized, and thereby get a head start in reaping the benefits. Visionaries are also likely to use the technology for specific applications, although the number of users may be significantly larger.

At this stage, incentive systems are again unnecessary as users of specific applications have implicit shared goals. There is also an inherent self interest for visionaries to see the technology that they choose succeed. Once there is a strong enough build up of visionaries and the technology has proven its worth, it is then possible to make the leap from the early market to the mainstream market, where the *pragmatists* await.

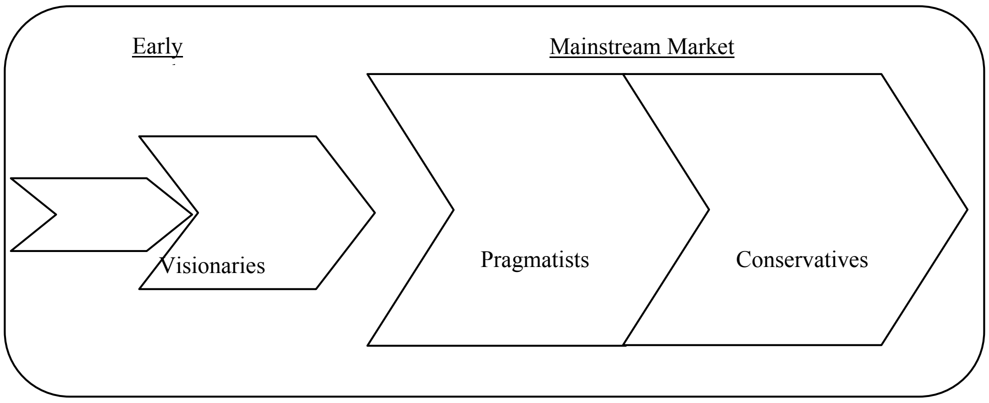
Pragmatists want a product that works and unlike the customers in the early market are much less tolerant of faults. They want to be able to buy products that meet their needs out of the box and easily get support from people who've used the technology before as well as find books about it in the bookstore. In short, they want a complete solution rather than a product that is still in development.

At this point of the technology's adoption, devices are reasonably ubiquitous and the technology has advanced beyond what was available in the early days. Most importantly, there are now a lot of experimental results and experience with real life implementations of the technology; it is also better understood how people actually use and abuse the system.

It is only at this point in the adoption cycle that it may make sense to introduce some form of incentives system. Even then, it would be better to design these incentives specifically for individual applications, based on what has been learned on how people abuse the networks, rather than a general incentives system that would possess the flaws discussed previously. As discussed in [18], it is unlikely that large-scale ad hoc networks will be deployed for general consumer applications due to their limitations in comparison to competing technologies. Their strengths will best be shown in either small-scale general applications or specific larger scale applications. In both cases, incentives can stem from common interest rather than an enforced system.

Finally, should mobile ad hoc networks become truly ubiquitous and used for general applications, c*onservatives* will hop onto the bandwagon, simply because they have no choice. Conservatives want products that are cheap and simple; they buy products only after everyone they know already owns one.

Figure shows the adoption cycle and the relative sizes of each customer segment.



- Figure 7.34. Adoption cycle for mobile ad hoc networks

It is of course still possible that in practice users will not wish to cooperate, even in the early stages of adoption. One of the main reasons behind the research of these systems is that the hardware and software of nodes can be tampered with and their behavior modified by users in such a way that the devices do not cooperate with others, in order to save resources. Although it is generally recognized that most users do not have the required level of knowledge and skills to modify nodes, there is concern that criminal organizations will have the resources and interest to produce and sell modified nodes on a large scale.

Our position is that the majority of users will only cheat when it is clearly beneficial to them and relatively easy to do. In the case of mobile ad hoc devices, it is unclear that there is significant benefit to be had from going to all the trouble to modify devices just to save resources such as battery power, memory and CPU cycles. Battery power is probably the most limited resource, and even that may not prove to be an issue to most users, as long as the devices do not require constant charges. Devices might also be designed with docking capabilities when the devices are stationary to reduce reliance on battery power as well as improve functionality. This will also encourage users to keep devices on to forward for others even when not in use. Also, if devices become truly ubiquitous the power needed for forwarding will decrease anyway as the distance from one hop to another becomes minimal.

While there certainly are plenty of criminal organizations with the ability to modify devices on a large scale, there is very little incentive for them to do so, since it is doubtful that a large enough market will exist to make the exercise profitable. A prominent example of a consumer device that has fallen victim to large scale tampering is the Sony Playstation 2 which has spawned an entire side industry of illegal modifications. Mod chips are widely available to buy on the Internet for home modification, as are full service organizations that modify units on behalf of consumers for a fee.

In the case of the Playstation, there are compelling reasons for both individual consumers and criminal organizations to engage in modification. Although the cost is relatively high, consumers who modify their devices can subsequently make significant savings by buying pirated games at a fraction of the original price. The organizations thus have a large and willing market of customers for their modifications, and are able to charge a significant sum to make large profits.

Conversely, in the case of mobile ad hoc devices, practically the only benefit to consumers would be longer battery lives. It is somewhat unlikely that they would go out of their way and pay a premium to modify their devices to this end, especially when it might cost the same to simply buy an extra battery with the added benefit of not voiding the device warranty or breaking the law. With little demand and potential for profitability, criminal organizations will not go to the trouble to reverse engineer and modify devices.

In any case, it would be necessary to produce a unique ad hoc device for each different type of application (e.g. a multiplayer ad hoc gaming device would be significantly different from an in-car ad hoc communications system). The need to reverse engineer each type of device as opposed to just one standard device would further increase costs and complexity for criminals and make it even less feasible for large scale modifications to occur.

To summarize, in this section by analyzing the cooperation issues in ad hoc network by taking into consideration the economic value for users to not cooperate we argued that there might not be a need for incentive systems at all, especially in the early stages of adoption, where excessive complexity can only hurt the technology's deployment. We looked at the needs of different customers segments for each stage within the projected technology adoption cycle and proposed that incentive systems not be used until ad hoc networks enter mainstream markets.

Even then, incentive systems should be tailored to the needs of each individual application rather than a general cookie cutter solution that may be too flawed or technically demanding to be implemented in reality. Punishments/incentives other than the denial of service to misbehaving nodes might be considered as an alternative. For example, within a file sharing application, users might be punished by limiting their query returns, rather than ostracizing them from the network completely.

History is littered with examples of great technologies that never saw the light of day due to deployments that attempted to achieve too much too fast, with no way of successfully monetizing the technology or build up acceptance; all of which are dangers that ad hoc networks face. It is important to remember that mobile ad hoc networks are only one of a host of competing technologies, and in order to successfully make it to the mainstream market its worth over competing technologies needs to be clear and proven to consumers.

An important caveat to note is that the problem of providing incentives to selfish nodes is a somewhat separate issue from preventing malicious attacks on the network. In this paper we have addressed the problem of nodes that wish to maximize their personal utility of the network, whereas malicious users may be less concerned with personal gain and simply wish to attack the network. Therefore, whilst we argue that incentive systems may not be necessary, it is still imperative that there are mechanisms to prevent against malicious attacks to maintain the reliability of the network.

In conclusion, it is unlikely that there is a perfect solution to the ad hoc incentives problem. Implementations of technology are always limited in reality by cost, human behavior, complexity and resources. Indeed, there is often only a least bad solution that provides the best cost benefit ratio rather than a best solution. It is more important at this point that mobile ad hoc networks be given the space to grow and develop than to choke it with complicated solutions to problems that may not even exist causing users to shun the technology.

7.4. References

[1] S. Bendahan, G. Camponov and Y. Pigneur, "Multi-issue Actor Analysis: Tools and modes for assessing technology environments," *Journal of Decisions Systems*, May 2003.

[2] L.Buttyan and J.P.Hubaux, "Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks" *ACM/Kluwer Mobile Networks and Applications*, Vol. 8, No. 5, October 2003.

[3] G. Camponovo and Y Pigneur, "Analyzing the M-Business Landscape," *Annals of Telecommunication*, 2002.

[4] B. Lamparter, K. Paul and D. Westhoff, "Charging Support for Ad Hoc Stub Networks," *Computer Communications*, Vol 26, pp 1504-1514, 2003.

[5] E.Mitjana and D. Wisely "Towards Scenarios and Business Models for the Wireless World," *Wireless World Research Forum WWRF#5*, Tempe, Arizona, Mar 2003.

[6] T.Robles, N. Umeda and G. Neureiter, "Service Scenarios and Business Models in BRAIN Project," *Proc. International Workshop on Broadband Radio Access for IP-based Networks*, London, Nov 2003.

[7] Raywood Communications, "Innovative GPS Job Dispatch and vehicle tracking systems" http://www.raywoodcomms.com.au/brochures/Dispatch.pdf

[8] Robert M. Hardaway, "Taxi and Limousines: The Last Bastion of Economic Regulation", *Hamline Journal of Public Law & Policy*, Spring 2000 http://madison.indymedia.org/newswire/display/2384/index.php

[9] Y. Cheng, "Singapore's Smart Taxis", *Asian Technology Information Program (ATIP),* March 1996 http://www.atip.org/public/atip.reports.96/atip96.016r.html

[10] John R. Stone, Gorman Gilbert, and Anna Nalevanko, "Assessment of Computer Dispatch Technology in the Paratransit Industry", *U.S. Department of Transportation* http://www.rapts.cutr.usf.edu/PDF/APTS%20Technologies/Fleet%20Managment%20Systems/Paratransit%20CAD%20Assessment%20Mar92.pdf (monetary figures have been converted to current day values using http://eh.net/hmit/ppowerusd/)

[11] Bj¨orn Knutsson, Honghui Lu, Wei Xu, Bryan Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games", *INFOCOM 2004*, March 2004 http://www.cis.upenn.edu/~hhl/Papers/infocom04.pdf

[12] Yanna Vogiazou, "Presence Based Massively Multiplayer Games: Exploration of a new concept", *Knowledge Media Institute*, April 2002 http://kmi.open.ac.uk/publications/papers/kmi-tr-123.pdf

[13] John Halloran, Yvonne Rogers & Geraldine Fitzpatrick, "From Text to Talk: Multiplayer Games and Voiceover IP", *Level Up: 1st International Digital Games Research Conference*, 2003 http://www.cogs.susx.ac.uk/users/johnhall/DiGRA03.pdf

[14] Markus Sabadello, "Small group multiplayer games", April/May 2001 http://www.cg.tuwien.ac.at/courses/Seminar/SS2001/multiplayer/small_group_multiplayer.pdf

[15] "True Mobile Multiplayer Games Arrive (press release)", Synergenix Interactive http://www.synergenix.se/downloads/Move_Pressrelease.pdf

[16] "White Paper - Massive multiplayer game networking", *Syncrotec & Norsk Regnesentral (The Norwegian Computing Centre)*, October 2001 http://www.nr.no/dart/projects/gisa/download/whitepaperGISAv5.pdf

[17] Loren Shuster, "Global Gaming Industry Now a Whopping $35 Billion Market", Compiler Magazine, July 2003 http://www.synopsys.com/news/pubs/compiler/art1lead_nokia-jul03.html

[18] "Video Game Industry", prepared by RocSearch, May 2003 http://www.rocsearch.com/pdf/Video%20Game%20Industry.pdf

[19] Eric Low, "Chaos in the Value Chain: Non-Traditional Paths to Market for Wireless Games", *GIGnews.com* http://www.gignews.com/biz/wirelessnov03.htm

[20] MobileMAN Project Presentation, IST Report IST-2001-38113, Dec 2002. http://cnd.iit.cnr.it/mobileMAN/deliverables/MobileMAN_D0.pdf

[21] L.Buttyan and J.P.Hubaux, August 2000, "Enforcing Service Availability in Mobile Ad-Hoc WANs" Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing

[22] L.Buttyan and J.P.Hubaux, January 2001, "Nuglets: a Virtual Currency to Stimulate Cooperation in Self Organized Mobile Ad Hoc Networks" Technical Report EPFL

[23] L.Buttyan and J.P.Hubaux, October 2003, "Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks" *ACM/Kluwer Mobile*

[24] C. Tschudin, 2003, "Embedding MANETs in the Real World" Conference on Personal Wireless Communications (PWC)

[25] S. Buchegger, J. Le Boudec, October 2002 "Cooperative Routing in Mobile Ad-hoc Networks: Current Efforts Against Malice and Selfishness" *Lecture Notes on Informatics, Mobile Internet Workshop, Informatik 2002*

[26] S. Buchegger, J. Le Boudec, 2002, "Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes – Fairness in Dynamic Ad-hoc NeTworks" *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing*

[27] P. Michiardi, R. Molva, 2002, "CORE: A Collaborative Repudiation Mechanism to enforce node cooperation in Mobile Ad hoc Networks" *Sixth IFIP conference on security communications, and multimedia (CMS 2002)*

[28] S. Zhong, Yang Richard Yang, J. Chen; Sprite, March 2003, "A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-hoc Networks" *Proceedings of INFOCOM 2003*

[29] E. Adar and B. A. Huberman, 2000 "Free riding on gnutella" First Monday, 5(10)

[30] M. Jakobsson, J.P. Hubaux, L. Buttyan, January 2003 "A micro-payment scheme encouraging collaboration in multi-hop cellular networks" *Proceedings of Financial Crypto 2003*

[31] Sorav Bansal and Mary Baker, 2003, "Observation based cooperation enforcement in ad hoc networks" Stanford University Technical Report

[32] H.-Y. Hsieh and R. Sivakumar, June 2001 "Performance comparison of cellular and multi-hop wireless networks: A quantitative study" *Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) 2001*

[33] B. Lamparter, K. Paul and D. Westhoff, 2003, "Charging Support for Ad Hoc Stub Networks," *Computer Communications*, Vol 26.

[34] L. Anderegg, S. Eidenbenz, 2003, "Ad hoc VCG: A Truthful and Cost-Efficient Routing Protocol for Mobile Ad hoc Networks with Selfish Agents", Proceedings of the 9th annual international conference on Mobile computing and networking (Mobicom)

[35] Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker, 2000 "Mitigating routing misbehaviour in mobile ad hoc networks" *Proceedings of MOBICOM 2000*

[36] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring, March 2003 " Modelling incentives for collaboration in Mobile Ad Hoc Networks", Proceedings of Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)

[37] Ross Anderson and Markus Kuhn, Nov 1996, "Tamper Resistance --- A Cautionary Note" USENIX Workshop on Electronic Commerce

[38] E. Huang, S. Östring, J. Crowcroft & I. Wassell, Jan 2004 "Developing Business Models for MobileMAN" T*echnical Report, Cambridge University Engineering Library*, CUED/FINFENG/TR.475

[39] Geoffrey A. Moore, 1991, "Crossing the Chasm" HarperBusiness

[40] Marc Bechler, Walter J. Franz and Lars Wolf, February 2003, "Mobile Internet Access in Fleetnet", Verteilten Systemen KiVS 2003, Leipzig, Germany

[41] E Huang, S Östring, J Crowcroft & I Wassell, "Developing Business Models for MobileMAN", Technical Report CUED/FINFENG/TR.475, Cambridge University Engineering Library, 2004 (omit for now)

[42] John R. Stone, Gorman Gilbert, and Anna Nalevanko, "Assessment of Computer Dispatch Technology in the Paratransit Industry", U.S. Department of Transportation (monetary figures have been converted to current day values using http://eh.net/hmit/ppowerusd/)
[43] Y. Cheng, "Singapore's Smart Taxis", Asian Technology Information Program (ATIP), March 1996
[44] Interview with with director of a local taxi company
[45] Dolphin Telecommunications homepage, http://www.dolphin-telecom.co.uk
[46] Cordic Communications website, www.cordic.com
[47] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu and Jorjeta jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols", in ACM/IEEE MobiCom, 1998
[48] Elizabeth M. Royer, P. Michael Melliar-Smith, and Louise E. Moser, "An analysis of the optimum node density for ad hoc mobile networks", IEEE ICC, June 2001
[49] Tracy Camp, Jeff Boleng and Vanessa Davies, "A Survey of Mobility Models for Ad Hoc Network Research", in Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol.2, no.5, 2002
[50] Fan Bai, Narayanan Sadagopan and Ahmed Helmy, "IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks." in IEEE INFOCOM, March 2003
[51] Christian Bettstetter, "Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks", MSWiM'01, July 2001
[52] Ben Liang and Zygmunt J. Haas, "Predictive Distance-Based Mobility Management for PCS Networks", in IEEE INFOCOM, March 1999
[53] Christian Schindelhauer, Tamás Lukovszki, Stefan Rührup and Klaus Volbert, "Worst Case Mobility in Ad Hoc Networks", in SPAA'03, June 2003
[54] Xiaoyan Hong, Taek Jin Kwon, Mario Gerla, Daniel Gu and Guangyu Pei, "A mobility framework for ad hoc wireless networks", in ACM International Conference on Mobile Data Management (MDM), Jan 2001
[55] Jungkeun Yoon, Mingyan Liu and Brian Noble, "Sound Mobility Models", in ACM/IEEE MobiCom, Sept 2003
[56] Simon R. Saunders, WILEY 1999, "Antennas and propagation for wireless communication systems"
[57] E Green, "Radio link design for microcellular systems", British Telecom Technology Journal Vol 8 No 1 Jan 1990
[58] Mattias Esbjornsson, Oskar Juhlin and Mattias Ostergren, Dec 2002, "The Hocman Prototype-fast motor bikers and ad hoc networking", International Conference on Mobile and Ubiquitous Multimedia
[59] Jorg Ott and Dirk Kutscher, May 2004, "The "Drive-thru" architecture: WLAN-based internet access on the road", IEEE Semiannual Vehicular Technology Conference
[60] Ingo Gruber, Oliver Knauf, Hui Li "Performance of Ad Hoc Routing Protocols in Urban Environments", Proceedings of European Wireless 2004 (EW'2004), Feb 2004
[61] Douglas M. Blough, Giovanni Resta and Paolo Santi, "A Statistical Analysis of the Long-Run Node Spatial Distribution in Mobile Ad Hoc Networks", in MSWiM'02, Sept 2002
[62] Christian Bettstetter, Hannes Hartenstein and Xavier Pérez-Costa, "Stochastic Properties of the Random Waypoint Mobility Model", in ACM/Kluwer Wireless Networks, Special Issue on Modeling & Analysis of Mobile Networks, Vol. 10, No. 5, Sept 2004
[63] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, Subhash Suri, "Towards Realistic Mobility Models For Mobile Ad hoc Networks", in ACM/IEEE MobiCom, Sept 2003

[64] Jungkeun Yoon, Mingyan Liu, Brian Noble, “Random Waypoint Considered Harmful”, in IEEE Infocom, April 2003

[65] David D. Clark, John Wroclawski, Karen R. Sollins, Robert Braden, “Tussle in Cyberspace: Defining Tomorrow’s Internet”, in ACM SIGCOMM, August 2002

[66] Kevin Fall, “A Delay-Tolerant Network Architecture for Challenged Internets”, in ACM SIGCOMM, August 2003

[67] J. Jetcheva, Y.C. Hu, S. PalChaudhuri, A. Saha, D. Johnson, “Design and Evaluation of a Metropolitan Area Multitier Wireless Ad Hoc Network Architecture”, in Mobile Computing Systems and Applications, 2003

[68] R. Patra S. Jain, K. Fall. Routing in a delay tolerant network. In Proceedings of ACM SIGCOMM, 2004.

[69] E. Royer and C-K Toh. A review of current routing protocols for ad-hoc mobile wireless networks. IEEE Personal Communications Magazine, pages 46-55, 1999.

[70] M. Balazinska and P. Castro. Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. In 1st International Conference on Mobile Systems, Applications, and Services (MobiSys), San Francisco, CA, May 2003.

[71] M. McNett and G. M. Voelker. Access and mobility of wireless pda users. Technicalreport, Computer Science and Engineering, UC San Diego, 2004.

[72] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In Proceedings of ACM Mobicom, 2004.

[73] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. Wireless Communications and Mobile Computing: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, 2(5):483-502, 2002.

[74] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. MobiCom, San Diego, CA, 2003.

[75] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. IEEE/ACM Transactions on Networking, 10(4):477-486, 2002.

[76] G. Sharma and R. R. Mazumdar. Scaling laws for capacity and delay in wireless ad hoc networks with random mobility. In Proceedings of ICC 2004. IEEE, 2004.

[77] J. Su, A. Chin, A. Popivanova, A. Goel, and E. de Lara. User mobility for opportunisticad-hoc networking. In Proceedings of WMCSA 2004, 2004.

[78] P. Juang, H. Oki, Y. Wang, M. Martonosi, L-S Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 96-107, 2002.

[79] T. Small and Z. J. Haas. The shared wireless infostation model - a new ad hoc networking paradigm (or where there is a whale, there is a way). In Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, pages 233-244,2003.

[80] P. Bremaud. Markov Chains, Gibbs Field, Monte Carlo Simulation and Queues. SpringerVerlag, 1999.

[81] F. Baccelli and P. Bremaud. Elements of Queuing Theory. Springer-Verlag, second edition,2003.

[82] J. F. C. Kingman. Inequalities in the theory of queues. J. Roy. Statistical Society, Series B, 32:102-110, 1970.

[83] B.Karp and H.T.Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In Proceedings of 6th Annual Internation Conference on Mobile computing and Networking(MobiCom 2000), Boston, MA, USA, pages 243-254, Feb 2000.

[84] Prosenjit Bose, Pat Morin, Ivan Sto jmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. Wireless Networks, 7(6):609-616, 2001.

[85] C.Komar and C. Ersoy. Location tracking and location based service using IEEE 802.11 WLAN infrastructure. European Wireless 2004, Barcelona Spain, Feb 2004.

[86] NS group at ISI. Ns 2 home page. World Wide Web, http://www.isi.edu/nsnam/ns/.

[87] Netos group at University of Cambridge. Landmark guided forwarding home page. World Wide Web, http://www.cl.cam.ac.uk/Research/SRG/netos/LGF/.

[88] Yongjin Kim, Jae-Joon Lee, and Ahmed Helmy. Modeling and analyzing the impact of location inconsistencies on geographic routing in wireless networks. SIGMOBILE Mob. Comput. Commun. Rev., 8(1):48-60, 2004.

[89] Antony LaMarca, Yatin Chawathe, Sunny Conolvo, Jeffery Hightower, Ian Smith, James Scott, Tim Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill Schilt. Place lab: Device positioning using radio beacons in the wild. Technical report, Intel Research, 2004.

[90] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00), pages 120-130, August 2000.

[91] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. IEEE Network Magazine, 15(6):30-39, November 2001.

[91] Elizabeth M.Royer and C. K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. IEEE Personal Communications Magazine, pages 46-55, April 1999.

[93] Karim Seada, Ahmed Helmy, and Ramesh Govindan. On the effect of localization errors on geographic face routing in sensor networks. In IPSN'04: Proceedings of the third international symposium on Information processing in sensor networks, pages 71-80. ACM Press, 2004.

[94] D. Son, A. Helmy, and B. Krishnamachari. The effect of mobility-induced location errors on geographic routing in ad hoc networks: Analysis and improvement using mobility prediction, 2004.

[95] L. Tang and M. Crovella. Virtual landmarks for the Internet. In Proceedings of Internet Measurement Conference 143-152, Miami Beach, FL., October 2003.

[96] P. F. Tsuchiya. The Landmark hierarchy: a new hierarchy for routing in very large networks. In SIGCOMM '88: Symposium proceedings on Communications architectures and protocols, pages 35-42. ACM Press, 1988.

[97] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computing. Computer Communication Review, 24(4):234-244, Oct 1994.

[98] C. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing. Feb 1999.

[99] K. Merchant, W. Hsu, H. Shu, C. Hsu and A. Helmy. Weighted way mobility model and its impact on ad hoc networks (poster). In Proceedings of ACM/IEEE MobilCom, 2004.

[100] M. Musolesi, S. Hailes, and C. Mascolo. An ad hoc mobility model founded on social network theory. In Proceedings of ACM MSWiM 2004.

[101] D. Aguayo, J. Bicket, S. Biswas, G. Judd and R. Morris. Link-level measurements from an 802.11b mesh network. In Proceedings of SIGCOMM 2004.

[102] M. Takai, J. Martin and R. Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. Proceedings of ACM MobiHoc, 2001.

[103] R. Gray, D. Kotz, C. Newport, N. Dubrovski, A. Fiske, J. Liu, C. Masone, S. McGrath, and Y. Yuan. Outdoor experimental comparison of four ad hoc routing algorithms. In Proceedings of ACM MSWiM 2004.

8. AD HOC NETWORKING AND INTERNET EVOLUTION

8.1. Introduction

The research activities performed in the MobileMAN project provide the bases for new networking paradigms (e.g., *Community networks*, *Accidental networking*) which we envisage as an important part in the Future Internet. Below, we will briefly survey these innovative ideas and new research directions to tackle the new research issues: community networks, accidental networking, etc. Addressing these issues is beyond the scope of the MobileMAN project, but internal projects has been set up by various project partners to start investigating this promising ideas.

We are starting to see applications that are self-organizing in communities. This organization operates at all levels, from human-to-human, to establish identity and trustworthiness, down to the forwarding of data on behalf of each other to provide mutual support in the absence of a provider. Today, these communities use the Internet for connectivity, on top of which they create specific overlays. Peer-to-peer applications, file sharing systems, content search, naming, and route finding are community specific and are typically overlayed on the Internet. The analysis of hardware evolution indicates that the storage capacities of small devices will soon vastly exceed their communication abilities. This motivates communities to use communication mechanisms that depart from the traditional end-to-end principle: obtaining popular content such as movie or song is easier from a neighboring small device (i.e., in a local community) than via the Internet. We believe that the current Internet system design is not well suited for such communities, especially when we extend the requirements for sharing resources down to the (possibly wireless) links themselves.
Disaster recovery, or less severe scenarios, calls for the creation of ad-hoc communities; Unix to Unix copy (UUCP) was a technology that allowed sporadically connected computers to exchange emails or news postings. UUCP made no assumptions about mobility. The Delay-Tolerant Network architecture identifies a class of scenarios called “challenged networks” and offers mechanisms to cope with these challenged environments in the Internet context. Examples of challenged networks are terrestrial mobile networks, exotic media networks, military ad-hoc Networks or Sensor/Actuator Networks. In the UK, many cities are trying to build local community networks based on meshed networks. Universities worldwide have similar project for campuses. In the north of Sweden, 802.11 Access Points have been installed on the snow mobiles of the Same (Lapp) nomadic people. They apply ad hoc opportunistic routing to exchange and convey information. A similar approach was used by motorbike couriers to deliver e-mail to and from Vietnamese schools. It may therefore take days to propagate information and there is never or seldom a direct link between communicating nodes. The information flow is similar to epidemic routing.

The Internet relies on the end-to-end principle and a structured allocation of IP addresses. This was very successful for building the scalable, world wide Internet we enjoy today: for example, IP addresses have a topological meaning, which is essential for the good operation of the backbone. However, in self-organized community networks these same ingredients are not well suited. Consider for example an ad-hoc meeting of a large number of users equipped with laptops. Connectivity between nodes is extremely transient. Normal IP addresses have no topological meaning, because users belong to different organizations and typically use non-routable, special addresses while participating in the ad-hoc meeting. If we use the current internet protocols, communication between such users requires user A to find the IP address of user B, then find an IP route to this destination address (using an ad-hoc routing protocol), then map the IP address of the next hop to a MAC address. This is all very complex in the presence of mobility and partial connectivity, has a significant processing and transmission overhead, resulting in very poor performance. In many cases it simply fails, particularly when connectivity is sporadic. A more appropriate alternative might be to communicate directly at the application layer, without using IP routing: user A would send a query to its neighbors asking for information about user B, who would then forward it until someone finds B.

Therefore, new communication architectures are called for, departing from the current internet protocol that takes into consideration the new challenges of community networking such as mobility, disconnectedness and ad-hocness. For example, instead of searching futilely for an end-to-end routed path, connectivity must be achieved logically and asynchronously, perhaps only at the application layer. Community networking should not replace the Internet but rather sidesteps it by addressing entirely new application patterns using new technologies for information sharing. However, the Internet remains a bridging technology between communities, and provides connectivity within the community infrastructure.

8.2. Communities and their communication

A community is a network structured around users with a common interest or common property. Such a property could be the proximity to each other, such as university lab, a condominium, or meeting participants. Communities with common interest could be parents of kids attending a common school, fans of Elvis Presley, ethnic groups, users of file sharing P2P programs, instant messaging, presence applications, etc. Communities can also be built using overlay networks, as well as meshed network using ad hoc wireless links.
Communities can be long lived (e.g. common interest groups) or short-lived (e.g. a set of drivers stuck in the same traffic jam). Communities develop their own knowledge, how to identify, trust and reach each other and how to share information, condensed from their situation. A given node can be involved in multiple communities. A community network maintains its own knowledge including that of the network relationships. Thus there is no need for global addressing.

Stanley Milgram at Yale University, formulated the question of what probability is that in a given set N of people, each member of N is connected to another member via links? Milgram's conducted a practical social experiment by mailing 160 letters to a set of randomly chosen people. In this letter he asked them to pass these letters to an unknown target person using only intermediaries known to one another.
Each person would pass the letter to a friend whom she thought might bring the letter closest to the target; until the letter eventually reached the target. Results suggested that the average length of such social chains is roughly six. This is called the small world phenomenon. Recent theoretical work has since extended the model to a wide range of nonsocial networks.
Similarly, community' network messages (or queries) could be forwarded from one node to the next according to local decisions based on node's interest and neighborhood. The query is forwarded to neighboring node based on the probability that this node knows the answer or knows a potential recipient that can make the most progress towards the answer.
Inside a community network we can envisage a trust model that operates as in the real world. For example, we can bootstrap trust using human contact - when two people meet and recognize each other they can vouch for each others devices. If there are tamper proof or interference free channels between the users' machines, this can be quite easily achieved. From then on, it is possible to form chains of trust through reputation and recommendation systems. The question is how well these operate in an asynchronous disconnected world.

One problem identified in ad-hoc network incentive systems and also in P2P systems is the establishment of unique and trustworthy identities - We believe that this can be solved with the “human -in-the-loop” approach described here.

8.3. The Accidental Networking Project

This is a research project developed jointly by CNR and SUPSI to investigate a novel communication paradigm named *Accidental networking*, where communication is no more a mandate, but can happen (and happens when needed) combining the mobility of ad hoc networks with the capabilities of future sensor networks, and human behavior. Nodes and networks communications happens because of the nodes mobility, and when this happens nodes may exchange the information they collected.
We envisage as a natural application field the support to epidemiologic studies (e.g., the dynamics of biological diffusion of diseases, contagious and contacts, as well as non-healthy habits and relationships between environment and health). People, equipped with advanced networking technologies, become bearers of the record of their contacts (with other people, environment, etc..), behaviors, biochemical and functional parameters and of environmental pollution allowing the specialists to quickly identify changes on health parameters, wrong behaviors and/or environmental pollution and easily reconstruct the contamination path.
As a person contracts a disease without realizing he/she can collect his log of contacts with other people or environments. This is done by means of a set of advanced networks (ad hoc and sensor), in which communications follow the "accidental networking paradigm": the mobility of nodes/networks is used for exploiting the communication paradigm. A person, wearing a sensor network (Body Area Network – BAN) collects information about his contacts and health status, still without realizing it, following the same path the epidemiological diffusion the bio-agents follow, or the logical sequence of his actions. His BAN records it and gives this information to the specialist for check and diagnostic purpose. The same way, the BAN can alert some other individual of the dangerous exposure of a certain area, or other types of alerts and communications, to prevent bad habits.

The accidental networking model, which was developed by referring to epidemiological studies, perfectly suits to several other interesting scenarios (people-based communication, urban, environmental and health monitoring), and opens challenges for extending the role of sensor and ad hoc networks in real life.

8.3.1. The accidental networking model

The accidental networking model is composed by three different elements: the body area network (BAN) of sensor nodes, the accidental communication, and the proactivity.
In the BAN, a set of sensors is distributed onto the body of a person, which communicate among them creating a body area network. The sensors collect information about the dynamics of contacts (person-person, person-environment, person-object, etc…) and distribute them to a central collector (still in the BAN) for several epidemiological and related studies. The sensor nodes are sensing devices with the capability of dealing with the collected information. These nodes can either move, or be static, and have minimal communication capabilities. With minimal communication functions, we intend that these nodes are not necessarily required to communicate with other sensor nodes. Furthermore, they could be "passive" to the communication process, delivering their information to the collector nodes when requested. They also have minimal storage, as they are not supposed to collect and maintain a large quantity of other nodes information. They can be able to adapt dynamically to prevailing circumstances, both environmental (e.g. increased sample frequency with increasing likelihood of flooding) and infrastructural (e.g. synchronization of sensor hibernation to conserve battery life, adaptation of network routing around different failure modes of nodes and links). Ideally, they can also derive

all the resources necessary for surviving (self-sufficing sensors), from the environment (for example: the battery power).
The whole BAN acts as a node of an ad hoc network: can communicate with the other BANs, sensors distributed in the environment or in some objects for obtaining the information in a direct way, whenever the person wearing the BAN gets in the transmission range of another person, or of some other sensors (accidental networking). The single-hop communication is just the first brick for a multi-hop communications, where people acts as relay for other people information.
Both the sensors and the BANs can communicate with the specialist and the environment and trigger its reaction to adapt and anticipate to the most wished configuration (proactivity). For example, if from the parameters of some personal BANs result that a certain environmental factor generated some unexpected reaction on certain individuals, the specialist can request some modification to mitigate the diffusion of this factor.

8.3.2. Challenges and Objectives

1. Define the BANs of sensor nodes and their characteristics.
2. Define protocols for accidental communication and BAN-to-BAN communication.
3. Define tool and strategies for storing and accessing, communicating, handling and integrating data in the BANs, with the characteristic of being:
    - Dynamic
    - Adaptive
    - Self-organising and efficient
    - Delay tolerant, but minimizing the latency
    - Error resilient
4. Define the framework for the proactivity.
5. Study the costs optimisation of this model in terms of:
    - Nodes introduction and management
    - Information management (replication, aggregation, compression, distribution, etc…)
    - Communication
    - Organisation

8.4. The Pilgrim Project

The Pilgrim project, which is a direct follow up of the MobileMAN project, is an internal project of University of Cambridge. Pilgrim is targeted at information sharing in an Ad Hoc world: Information sharing, asynchronous "instant" messaging, and people/object location for large meetings. The Pilgrim system combines the familiar functionality of the Web through a fully decentralized indexing process, distributed search, and cross-layered peer-to-peer with ad-hoc networking.
It will enable a wide range of new applications that can be related to the notion of context. For example, a node context could be used to know who was around a given node at a given time to find witnesses for a car crash. Similarly, when one enters a store, the node can communicate with objects in the store to check availability, freshness, record a purchase date, etc. P2P systems, such as Freenet and ad-hoc routing systems are related to Pilgrim. Despite similar characteristics, they have a fundamental difference: they do not support intermittent connectivity. Tuple spaces have been proposed to support information sharing in mobile infrastructures such as LIME. However, it assumes plentiful communications resource.

8.4.1. Usage Scenarios, Applications, and their Requirements

When attending a conference, one often wants to make arrangements to meet with a set of colleagues. We could envisage not just traditional location services, but more direct linkage via all the attendees' mobile communications devices, thus provisioning of a set of newer applications including:

File sharing
How often in a meeting do you want a copy of someone else's presentation? It would be interesting to explore how existing mechanisms for peer-to-peer optimization of bandwidth can make use of a decentralized communications environment with intermittent connectivity, in addition to the decentralized storage environment that is such systems already use.

Instant messaging
Creating off-the-cuff meetings through messaging is familiar to users of SMS. Doing this without any infrastructure is another application of community networks.

Email
Can we construct an SMTP gateway, run on localhost, that sends messages via an ad-hoc method if its available and the recipient is "nearby", falling back on using the net if not (and, perhaps, using both at once to achieve speed/redundancy (but only being displayed once at the recipient's end).

Asynchronous Web Services
My browser sends a request ("HTTP GET") using the community network, which is diffused through the network by exploiting users that are close to me. Some time later, passing another person, the response is delivered and my browser picks it up and I switch back to that thread of discussion or thought.

8.4.2. The Pilgrim Architecture for Community Networks

Pilgrim is made of three interlocking elements comprising the information model, the communications model and the security model. Each of these elements has a number of components, some shared, described next.

The Information Model

Each node builds its own connectivity database inside a community by building a set of relationships. These relationships are then used to find the "best" way (or next hop) to reach a given destination, as illustrated in the small world model.

The Communications Model

There are major differences between Pilgrim and the Internet:

(i) The choice of the next node relies on a local knowledge database.

(ii) The system will use a decentralized naming service. Nodes must be identified in each community. Names and addressed can be different for each community depending on the semantic of the community. Message forwarding is then also local or transformed into the forwarding principles of the community network. The Application is the Network, and the namespace used for applications is also used for proximity and other neighbor information.

(iii) The legacy route concept does not exist anymore. A query is propagated through a set of cooperating nodes trying to find the nearest match to the query. A response might be emitted by someone simply because they have a better value for the attribute, and might not even reach the originator of the first content/attribute since it might not be reachable any more, or might already have a better answer.

Pilgrim must be able to use the Internet to forward a query when the Internet is the best way to forward a message. How to reach nodes beyond the community and how to be reached from outside the community is similar in principle to NAT.

The Security Model

Pilgrim requires a novel security architecture that involves several novel components:

Membership, Roles and Identity
A user can be part of multiple communities. Within different communities, a device might have different roles, preferences, and priorities. Membership of a community has to be controlled. Graceful joining and leaving of communities must be done in a distributed fashion without compromising security, reliability and efficiency since a central infrastructure may not be available.

Authenticity and privacy
A community needs to ensure privacy and authentication for each member without relying on a central Key administration.

Trustworthiness, Reputation and Cooperation
Users in communities have implicit or explicit qualities in their knowledge and performance. Trust, reputation and ability to deliver are important characteristics besides knowledge to build qualities. Mechanisms to build confidence between users and how to represent them as relations

are important design goals. The system discourages free-riders through the use of incentives, aligned with trust and reputation.

8.5. References

[1] I. Chlamtac, M. Conti, J.J.-N. Liu, "Mobile ad hoc networking: imperatives and challenges", *Ad Hoc Networks*, vol. 1, no. 1, Jul. 2003, pages: 13-64.
[2] R. Karrer, A. Sabharwal, E. Knightly, "Enabling Large-Scale Wireless Broadband: The Case for TAPs", *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, Jan. 2004, pages: 27-34.
[3] E. Baccarelli, M. Biagi, R. Bruno, M. Conti, E. Gregori, "Broadband Wireless Access Networks: a Roadmap on Emerging Trends and Standards" in *Broadband Services: Business Models and Technologies for Community Networks*, I. Chlamtac, A. Gumaste, C. Szabo (Editors), *John Wiley & Sons* (Publisher), Mar. 2005.
[4] C. Eklund, R.B. Marks, K.L. Stanwood, S. Wand, "IEEE Standard 802.16: A Technical Overview of the WirelessMAN^{TM} Air Interface for Broadband Wireless Access", *IEEE Communications Magazine*, vol. 40, no. 6, Jun. 2002, pages: 98-107.
[5] P. Gupta, P.R. Kumar, "The Capacity of Wireless Networks", *IEEE Transactions on Information Theory*, vol. 46, no. 2, March 2000, pages: 388-404.
[6] H Bölcskei, A.J. Paulraj, K.V.S. Hari, R.U. Nabar, W.W. Lu, "Fixed Broadband Wireless Access: State of the Art, Challenges, and Future Directions", *IEEE Communications Magazine*, vol. 39, no. 1, Jan. 2001, pages: 100-108.
[7] X. Quin, R. Berry, "Exploiting multiuser diversity for medium access control in wireless networks", in *Proceedings* of *IEEE Infocom 2003*, San Francisco, CA, 30 March-3 April 2003, vol. 2, pages: 1084-1094.
[8] R. Draves, J. Padhye, B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks", in *Proceedings* of *ACM MobiCom'04*, Philadelphia, PE, Sept. 26-Oct. 1, 2004, pages: 114-128.
[9] C.A. Santivanez, B. McDonald, I. Stavrakakis, R. Ramanathan, "On the scalability of ad hoc routing protocols", in *Proceedings* of *IEEE Infocom 2002*, New York, NY, June 23-27 2002, vol. 3, pages: 1688-1697.
[10] V. Gambiroza, B. Sadeghi, E. Knightly, "End-to-End Performance and Fairness in Multihop Wireless Backhaul Networks", in *Proceedings* of *MobiCom'04*, Philadelphia, PA, Sept. 26-Oct. 1, 2004, pages: 287-301.

9. Appendix

In this Appendix we'll introduce in a more formal way some basic concepts related to repeated games and infinitely repeated games. We will then show the definition of a *strategy* for a player and explain how to verify if a (simple) strategy is equilibrium for a game.

Repeated games theory

Consider a game *G* (which we'll call the *stage game* or the constituent game). Let the player set be $I=\{1,\ldots,n\}$. In our present repeated-game context it will be clarifying to refer to a player's stage game choices as actions rather than strategies. (We'll reserve "strategy" for choices in the repeated game). So each player has a *pure-action* space A_i. The space of action profiles is $A = X_{i\in I} A_i$. Each player has a von Neumann-Morgenstern utility function defined over the outcomes of *G*, $g_i : A \to \Re$, that in the particular case of the two players PD game takes the form of a payoff matrix as in Table 3.
Let *G* be played several times (perhaps an infinite number of times) and award each player a payoff which is the (discounted) sum of the payoffs she got in each period from playing *G*. Then this sequence of stage games is itself a game: a *repeated game*.
Two statements are implicit when we say that in each period we're playing the same stage game: a) for each player the set of actions available to her in any period in the game *G* is the same regardless of which period it is and regardless of what actions have taken place in the past and b) the payoffs to the players from the stage game in any period depend only on the action profile for *G* which was played in that period, and this stage-game payoff to a player for a given action profile for *G* is independent of which period it is played. Statements a) and b) are saying that the environment for our repeated game is *stationary* (or, alternatively, independent of time and history). This does not mean the actions themselves must be chosen independently of time or history.
We'll limit our attention here to cases in which the stage game is a one-shot, simultaneous-move game. Then we interpret a) and b) above as saying that the payoff matrix is the same in every period. We make the typical "*observable action*" or "*standard private monitoring*" assumption that the play which occurred in each repetition of the stage game is revealed to all the players before the next repetition. Therefore even if the stage game is one of imperfect information (as it is in simultaneous-move games)—so that during the stage game one of the players doesn't know what the others are doing/have done that period—each player does learn what the others did before another round is played. This allows subsequent choices to be conditioned on the past actions of other players. We'll see later in the paper that if we make the assumption of "*imperfect private monitoring*" results can be significantly different.
Before we can talk about equilibrium strategies in repeated games, we need to get precise about what a *strategy* in a repeated game is. We'll find it useful when studying repeated games to consider the semi-extensive form. This is a representation in which we accept the normal-form description of the stage game but still want to retain the temporal structure of the repeated game.
Let the first period be labeled $t=0$. The last period, if one exists, is period T, so we have a total of $T+1$ periods in our game. We allow the case where $T=\infty$, i.e. we can have an infinitely repeated game.
We'll refer to the action of the stage game *G* which player *i* executes in period *t* as a_i^t. The action profile played in period *t* is just the n-tuple of individuals' stage-game actions:

$$a^t = \left(a_1^t,\ldots,a_n^t\right) \qquad \text{(A.19)}$$

We want to be able to condition the players' stage-game action choices in later periods upon actions taken earlier by other players. To do this we need the concept of a *history*: a description of all the actions taken up through the previous period. We define the history at time t to be:

$$h^t = \left(a^0, a^1, \ldots, a^{t-1}\right) \tag{A.20}$$

In other words, the history at time t specifies which stage-game action profile (i.e., combination of individual stage-game actions) was played in each previous period. Note that the specification of h^t includes within it a specification of all previous histories $h^0, h^1, \ldots, h^{t-1}$. For example, the history h^t is just the concatenation of h^{t-1} with the action profile a^{t-1}; i.e. $h^t = (h^{t-1};a^{t-1})$. The history of the entire game is $h^{T+1}=(a^0,a^1,\ldots, a^T)$. Note also that the set of all possible histories h^t at time t is just:

$$A^t = \mathop{\Large X}_{j=0}^{t-1} A, \tag{A.21}$$

the t-fold Cartesian product of the space of stage-game action profiles A.
To condition our strategies on past events, then, is to make them functions of history. So we write player i's period-t stage-game strategy as the function s_i^t, where $a_i^t = s_i^t(h^t)$ is the stage-game action she would play in period t if the previous play had followed the history h^t. A player's stage-game action in any period and after any history must be drawn from her action space for that period, but because the game is stationary her stage-game action space A_i does not change with time. The period-t stage game strategy profile s^t is:

$$s^t = \left(s_1^t, \ldots, s_n^t\right) \tag{A.22}$$

So far we have been referring to stage-game strategies for a particular period. Now we can write, using these stage-game entities as building blocks, a specification for a player's strategy for the repeated game. We write player i's strategy for the repeated game as:

$$s_i = \left(s_i^0, s_i^1, \ldots, s_i^T\right) \tag{A.23}$$

i.e. a (T+1)-tuple of history-contingent player-i stage-game strategies. Each s_i^t takes a history $h^t \in A^t$ as its argument. The space S_i of player-i repeated-game strategies is the set of all such (T+1)-tuples of player-i stage game strategies $s_i^t : A^t \to A_i$.
We can write a strategy profile s for the whole repeated game in two ways. We can write it as the n-tuple profile of players' repeated-game strategies:

$$s = \left(s_1, \ldots, s_n\right) \tag{A.24}$$

as defined in (A.23). Alternatively, we can write the repeated-game strategy profile s as:

$$s = \left(s^0, s^1, \ldots, s^T\right) \tag{A.25}$$

i.e., as a collection of stage-game strategy profiles, one for each period, as defined in (A.22).

Let's see how this repeated game is played out once every player has specified her repeated-game strategy s_i. It is more convenient at this point to view this repeated-game strategy profile as expressed in (A.25), i.e. as a sequence of T+1 history-dependent stage-game strategy profiles. When the game starts, there is no past play, so the history h^0 is degenerate: every player executes her $a_i^0 = s_i^0$ stage-game strategy from (A.23). This zero-th period play generates the history $h^1=(a^0)$, where $a^0 = \left(a_1^0,...,a_n^0\right)$. This history is then revealed (or monitored by the players themselves) to the players so that they can condition their period-1 play upon the period-0 play. Each player then chooses her *t=1* stage-game strategy $s_i^1(h^1)$. Consequently, in the t=1 stage game the strategy profile $a^1 = s^1(h^1) = \left(s_1^1(h^1),...,s_n^1(h^1)\right)$ is played. In order to form the updated history this stage-game strategy profile is then concatenated onto the previous history: $h^2=(a^0,a^1)$. This new history is revealed to all the players and they each then choose their period-2 stage-game strategy $s_i^2\left(h^2\right)$, and so on. We say that h^{T+1} is the path generated by the repeated-game strategy profile s.

Let us now consider the *payoff function of the repeated game.* We can think of the players as receiving their stage-game payoffs period-by-period. Their repeated game payoffs will be an additively separable function of these stage-game payoffs. Right away we see a potential problem: if the game is played an infinite number of times, there is an infinite number of periods and, hence, of stage-game payoffs to be added up. In order that the players' repeated-game payoffs be well defined we must ensure that this infinite sum does not blow up to infinity. We ensure the finiteness of the repeated-game payoffs by introducing *discounting* of future payoffs relative to earlier payoffs. Such discounting can be an expression of time preference and/or uncertainty about the length of the game. We introduce the average discounted payoff as a convenience which normalizes the repeated-game payoffs to be "on the same scale" as the stage game payoffs.
Infinite repetition can be the key for obtaining behavior in the stage games which could not be equilibrium behavior if the game were played once or a known finite number of times. For example, defection in every period by both players is the unique equilibrium in any finite repetition of the PD[38]. When repeated an infinite number of times, however, cooperation in every period is an equilibrium if the players are "sufficiently patient".
When studying infinitely repeated games we are concerned about a player who receives a payoff in each of infinitely many periods. In order to represent her preferences over various infinite payoff streams we want to meaningfully summarize the desirability of such a sequence of payoffs by a single number. A common assumption is that the player wants to maximize a weighted sum of her per-period payoffs, where she weights later periods less than earlier periods. For simplicity this assumption often takes the particular form that the sequence of weights forms a geometric progression: for some fixed $\delta \in (0,1)$, each weighting factor is δ times the previous weight. δ is called her *discount factor*. If in each period *t* player *i* receives the payoff u_i^t, we could summarize the desirability of the payoff stream $u_i^0, u_i^1,...$ by the number:

$$\sum_{t=0}^{\infty} \delta^t u_i^t \qquad \text{(A.26)}$$

[38] See theorem 4 in "Repeated Games" handouts by J. Ratliff [10].

Such an intertemporal preference structure has the desirable property that the infinite sum of the weighted payoffs will be finite (since the stage-game payoffs are bounded). A player would be indifferent between a payoff of x^t at time *t* and a payoff of $x^{t+\tau}$ received τ periods later if:

$$x^t = \delta^\tau x^{t+\tau} \tag{A.27}$$

A useful formula for computing the finite and infinite discounted sums we will use later in this section is:

$$\sum_{t=T_1}^{T_2} \delta^t = \frac{\delta^{T_1} - \delta^{T_2+1}}{1-\delta} \tag{A.29}$$

which, in particular, is valid for $T_2 = \infty$.

If we adopted the summation (A.26) as our players' repeated-game utility function, and if a player received the same stage-game payoff v_i in every period, her discounted repeated-game payoff, using (A.29), would be $v_i/(1-\delta)$. It is however more convenient to transform the repeated-game payoffs to be "on the same scale" as the stage-game payoffs, by multiplying the discounted payoff sum from (A.26) by $(1-\delta)$. So we define the average discounted value of the payoff stream $u_i^0, u_i^1, \ldots$ by:

$$(1-\delta)\sum_{t=0}^{\infty} \delta^t u_i^t \tag{A.30}$$

It is often convenient to compute the average discounted value of an infinite payoff stream in terms of a leading finite sum and the sum of a trailing infinite substream. For example, say that the payoffs v_i^t a player receives are some constant payoff v_i' for the first *t* periods, i.e. 0,1,2,…,*t*-1, and thereafter she receives a different constant payoff v_i'' in each period *t*,*t*+1,*t*+2,…. The average discounted value of this payoff stream is:

$$(1-\delta)\sum_{\tau=0}^{\infty} \delta^\tau v_i^\tau = (1-\delta)\left(\sum_{\tau=0}^{t-1} \delta^\tau v_i^\tau + \sum_{\tau=t}^{\infty} \delta^\tau v_i^\tau\right) = (1-\delta)\left(\frac{v_i'(1-\delta^t)}{1-\delta} + \frac{v_i''\delta^t}{1-\delta}\right) = (1-\delta^t)v_i' + \delta^t v_i'' \tag{A.31}$$

It is possible to see that the average discounted value of this stream of bivalued stage-game payoffs is a convex combination of the two stage-game payoffs. We can iterate this procedure in order to evaluate the average discounted value of more complicated payoff streams. Another useful example is when a player receives v_i' for the first *t* periods, then receives v_i'' only in period *t* and receive v_i''' every period thereafter. The average discounted value of the stream beginning in period *t* (discounted to period *t*) is: $(1-\delta)v_i'' + \delta v_i'''$. Substituting this for v_i'' in (A.31), we find that the average discounted value of this three-valued payoff stream is:

$$(1-\delta^t)v_i' + \delta^t\left[(1-\delta)v_i'' + \delta v_i'''\right] \tag{A.32}$$

We have now defined all the formalism needed to examine the equilibrium of a (infinitely) repeated PD game and to verify if a predefined strategy constitutes an equilibrium. The various definitions of equilibrium and the related theorems can be found in: J. Ratliff, Game Theory Handouts available at http://virtualperfection.com/gametheory.