MOBILEMAN

IST-2001-38113

Mobile Metropolitan Ad hoc Networks

MobileMAN Presentation

Deliverable D15

Contractual Report Preparation Date: July 2005 (*)
Actual Date of Delivery: 29 July 2005
Estimated Person Months: 7.5
Number of pages: 149

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

Authors: Marco Conti, Franca Delmastro, Enrico Gregori, Antonio Pinizzotto (CNR), Jon Crowcroft, Andrea Passarella (Cambridge), Claudio Lavecchia, Pietro Michiardi, Refik Molva (Eurecom), Jose Costa Requena, Mohammand Ayyash (HUT), Piergiorgio 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 report on the international workshop we organized to disseminate the MobileMAN activities inside the scientific and industrial community. To this end, in the framework of the *IEEE International Conference on Pervasive Services 2005 (ICPS'05)* we organized the First IEEE ICPS Workshop on *Multi-hop Ad hoc Networks: from theory to reality*, REALMAN 2005 (http://www.cl.cam.ac.uk/realman)

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

Summary

The REALMAN workshop idea steams from the lessons we learnt in the framework of the MobileMAN project. The results obtained in the MobileMAN project and the emerging world-wide literature indicated that to consolidate the ad hoc networking field we need to complement theoretical research activities with the realization and testing of realistic small/medium scale testbed. Indeed, after almost a decade of research into ad hoc networking, MANET technology has not yet affected our way of using wireless networks and there are no clear results showing how well MANETs work in reality as the research in this area has been based on theoretical analyses and simulation results only. Simulation models often introduce simplifications and assumptions that mask (in simulation experiments) important characteristics of the real protocols behavior. To make mobile multi-hop ad hoc networks (MANETs) a reality, simulation modeling and theoretical analyses have to be complemented by real experiences (e.g., measurements on real prototypes) which provide both a direct evaluation of ad hoc networks and, at the same time, precious information for a realistic tuning of simulation models. In addition, the availability of prototypes will also make possible to start creating communities of MANET users that, by experimenting with this technology, will provide feedbacks on its usefulness and stimulate the development of applications tailored for the ad hoc environment. This is fundamental to reduce the gap between what end users might find useful, and what research is currently addressing, making the cost of using ad hoc networking lower than the potential benefit.

The need for more experimental activities has stimulated the emerging of a new community of researchers combining theoretical research on ad hoc networking with experiences/measurements obtained by implementing ad hoc network prototypes. The aim of REALMAN was to bring together this community and to disseminate the MobileMAN results to the scientific community. The workshop constituted a unique forum for presenting and discussing experiences/results from real ad hoc networks test-beds and prototypes.

The answer to the open call for papers was very encouraging. We received 39 submissions out of which we selected 13 papers for presentation in the workshop sessions. In addition, in response to a separate call we received several interesting

demo proposals. The final program also included 8 demos (5 from MobileMAN partners) and 4 posters. 36 worldwide researchers participated to the workshop. These numbers show that REALMAN is already a reference forum for the growing community of researchers working in this field.

The workshop also included a panel "How to make MANETs scale and provide coverage *for real*" to discuss/identify research directions for making mobile ad hoc networks a reality. The panel, organized and chaired by Jon Crowcroft, had a set of panelists -- Matthias Grossglauser (*EPFL*), Edward Knightly (*RICE University*), Martin Mauve *(Duesseldorf University),* Joerg Ott *(Helsinki University of Technology*), Christian Tschudin (*Basel University*) -- that addressed the discussion from two different perspectives. Specifically, Matthias Grossglauser started by observing that applications are hard to predict (and hence scale is easy to underestimate); he pointed out the value of model-based research, especially for architectural decisions. On the opposite, the other speakers emphasized the key role of applications and real implementations. Martin Mauve took the opposing view to Matthias Grossglauser: application led research is better! Martin gave many examples from a large vehicle/car-to-car network they built, telling how the wireless network design worked much better when the architecture was application centered.

Edward Knightly and Joerg Ott discussed two directions for the evolution of ad hoc networks: mesh networks and opportunistic networking. Edward Knightly discussed research challenges in implementing a real and scalable mesh network by exploiting his experience with the RICE TAP project; Joerg Ott talked about opportunistic networking and pointed out very nice real examples to show that opportunistic networking (i.e., delay tolerant networks, DTN) can immediately provide useful applications for ad hoc networking. In Joerg view DTN precedes MANET.

Christian Tschudin summarized the discussion pointing out the need for a pragmatic approach in the mobile ad hoc research: use an application driven approach, and integrate/consolidate existing algorithms and software. Specifically, he stated: *While in theory it might be feasible to have 100 hop paths, I don't think that this will happen in real MANETs. The limit lies somewhere below and is determined more by pragmatic property bundles like routing stability or TCP fairness rather than single functions (finding a route) or optimizations (battery life time). What can theory,*

what can systems work contribute to bracket this ad hoc horizon? Agreeing on the "real objectives" would be a first step.

The slides of the panel presentations and other workshop material can be found at: http://www.cl.cam.ac.uk/realman

The workshop proceedings are in the attached Appendix.

Appendix

This Appendix contains the referred proceedings of the First IEEE ICPS Workshop on *Multi-hop Ad hoc Networks: from theory to reality*, REALMAN 2005 (http://www.cl.cam.ac.uk/realman)

REAL MAN 05

Santorini, Greece
July 14th, 2005

Proceedings of the

IEEE ICPS Workshop on
Multi-hop Ad hoc Networks: from theory to reality

The workshop is jointly organised by:

UNIVERSITY OF CAMBRIDGE

Mobile MAN

IIT Istituto di Informatica e Telematica del CNR

MobileMAN (IST-2001-38113) Project
funded by the FET-IST Programme

Information Society Technologies

Table of Contents

Session III: Deploying MANET Test-beds

Thoughts on Mobile Ad-hoc Network Testbeds
Wolfgang Kieβ, Stephan Zalewski, Andreas Tarp, and Martin Mauve

Experiences Deploying an Ad-hoc Network in an Urban Environment
Peter Barron, Stefan Weber, Siobhán Clarke, and Vinny Cahill

MeshDV: A Distance Vector mobility-tolerant routing protocol for Wireless Mesh Networks
Luigi Iannone and Serge Fdida

Posters and Demos Session

Social networks, novel communication applications and needs in mobile contexts
Claudia Brazzola

On the Dimensionality of Wireless Connectivity Traces
George Roussos

Cross-Layer Support for Group-Communication Applications in MANETs
Marco Conti, Franca Delmastro, Jon Crowcroft, and Andrea Passarella

Real Life Experience of Cooperation Enforcement Based on Reputation (CORE) for MANETs
Claudio Lavecchia, Pietro Michiardi, and Refik Molva

VoIP Testbed in Ad Hoc Networks
Jose Costa-Requena, Mohammand Ayyash, Jarrod Creado, Jarkko Hakkinen, Raimo Kantola, and Nicklas Beijar

Experimenting a Layer 2-based Approach to Internet Connectivity for Ad Hoc Networks
Raffaele Bruno, Marco Conti, Enrico Gregori, Antonio Pinizzotto, and Emilio Ancillotti

Demo of Ana4: an Hybrid Local Area Ad hoc Network Architecture
Nicolas Boulicault, Guillaume Chelius, and Eric Fleury

Haggle Architecture and Demo of its Real World Implementations
Pan Hui, Jon Crowcroft, James Scott, Christophe Diot, Augustin Chaintreau, and Richard Gass

Demo of residual bandwidth estimation in an 802.11 ad hoc network
Martin Nielsen

Organizing Committee

General Chair: Jon Crowcroft, University of Cambridge, UK

Program Chair: Marco Conti, National Research Council, IIT Institute, Italy

Program Vice-Chair: Andrea Passarella, University of Cambridge, UK

Technical Program Committee:
G. Anastasi, Pisa University, Italy
A.T. Campbell, Columbia University, USA
S.R. Das, SUNY at Stony Brook, USA
A. Ephremides, University of Maryland, USA
S. Giordano, SUPSI, CH
E. Gregori, IIT-CNR, Italy
T. Henderson, Dartmouth College, USA
R. Kantola, Helsinki University of Technology, Finland
H. Karl, Paderborn University, Germany
E. Knightly, Rice University, USA
M. Mauve, Dusseldorf University, Germany
P. Michiardi, EURECOM, France
R. Molva, EURECOM, France
S. Olariu, Old Dominion University, USA
G. Polyzos, AUEB, Greece
D. Remondo, Catalonia University of Technology, Spain
C. Rohner, Uppsala University, Sweden
J.H. Schiller, Freie University Berlin, Germany
J. Scott, Intel Research, UK
F. Sestini, European Commission
V.A. Siris, FORTH-ICS and Crete University, Greece
I. Stavrakakis, Athens University, Greece
I. Stojmenovic, Ottawa University, Canada
C. Tschudin, Basel University, Switzerland
N. Vaidya, University of Illinois at Urbana-Champaign,USA
J. Wu, Florida Atlantic University, USA

SESSION I

MANET Implementations

Experiments with an enhanced MAC architecture for multi-hop wireless networks

R. Bernasconi[1], R. Bruno[2], I. Defilippis[1], S. Giordano[1,2], A. Puiatti[1]
1: University of Applied Sciences, (SUPSI), Department of Technology and Innovation (DTI), 6928 Manno, Switzerland
{ralph.bernasconi, ivan.defilippis, silvia.giordano, puiatti}@.supsi.ch
2: National Research Council, (CNR), IIT Institute, 56100 Pisa, Italy
raffaele.bruno@iit.cnr.it

Abstract

In this paper we describe the architecture of a wireless network interface card that we have used for implementing an enhanced IEEE 802.11 MAC protocol for multi-hop wireless networks, which adopts a backoff mechanism, called Asymptotically Optimal Backoff (AOB) [5], specifically designed for high contention conditions. We have performed traffic experiments with this enhanced card to compare it against the traditional IEEE 802.11 MAC protocol. The results of our experiments, mainly devoted to conflict and interferences situations, show that our enhanced version of the IEEE 802.11 MAC protocol is more robust to both high contention and interferences situations than the standard one, resulting in a significant improvement of per-station throughput performance[1].

1. Introduction

Instantly deployable, multi-hop wireless networks can be both fully self-organized (ad hoc networks) and connected to the wired backbone via multiple wireless hops (multi-hop wireless LANs). Potential applications of this type of networks include rescue, citizen and military networks, where users self-organize to communicate or to access the Internet; and multi-hop Hot Spots, i.e., wireless networks in public areas where traditional WLAN technology is augmented with ad hoc wireless communications.

Today, the *de facto* technology for building multi-hop ad hoc networks is the IEEE 802.11 standard [1]. However, it is extensively recognized that the 802.11 standard 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 [2-6]. Several feedback-based mechanisms have been proposed to maximize the IEEE 802.11 MAC protocol efficiency by guaranteeing an *optimal* time spreading of the users' access [2], [4-6]. Specifically, in [5] the *Asymptotically Optimal Backoff (AOB)* mechanism was proposed, which dynamically tunes the backoff parameters such 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.

In a previous paper [14], we proposed the architecture of an enhanced IEEE 802.11 wireless network interface card, which is capable of seamlessly working in a multi-hop wireless network, but, at the same time, it is still fully compatible with traditional IEEE 802.11 implementations. This Medium Access platform has been designed to be a suitable architecture for implementing and testing: 1) backoff algorithms more adequate to multi-hop operations; 2) dynamic channel switching schemes to exploit channel quality diversity; 3) efficient packet forwarding schemes inside the MAC layer; and 4) cross-layering optimizations through the exploitations of topology information provided by the routing layer.

In this paper, we present our activity concerning the point 1) above. Specifically, we describe our first card implementation, with the introduction of the AOB scheme as described in [5]. We decided to implement the AOB algorithm, because it relies only on topology-blind estimates of the network status based on the standard physical carrier sensing activity. Hence, it appears as a suitable solution for multi-hop configurations. The experimental results obtained comparing our enhanced MAC card with traditional IEEE 802.11

[1] This work was partially funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-38113 MOBILEMAN project.

wireless cards, show the significant per-station throughput improvement ensured by the enhanced MAC protocol. Furthermore, they open promising directions to investigate additional enhancements, as discussed in Section 4.

The rest of this paper is organized as follows: in Section 2 we outline our implementation and the main hardware and firmware design choices. In Section 3 we describe the measurement environment and we present the results of our real experiments, discussing the most relevant points. Section 4 concludes this paper with some further discussion and detailed description of the ongoing and future work.

2. Overview of the Wireless Network Interface (WNI)

The Wireless Network Interface (WNI) is a multi-component system that has been designed to implement and test enhanced MAC protocols. It is portable, and therefore suitable for mobile use, provided that the required power can be supplied.

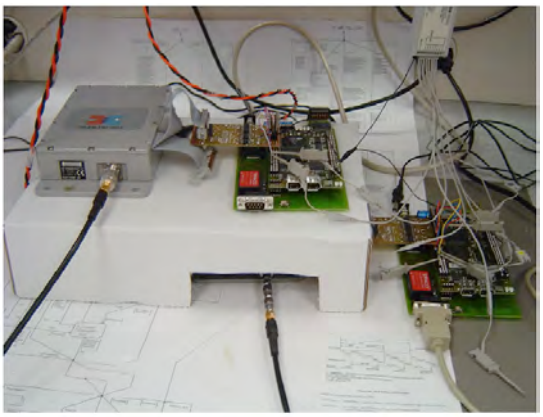


Figure 1: The enhanced IEEE 802.11 Wireless Network Interface (WNI).

The system consists of Radio Frequency up/down converters (RF), the Baseband modulator/demodulator (BB), and the MAC firmware on a compact DSP microcontroller.

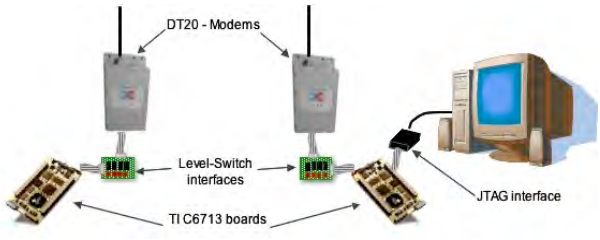


Figure 2: Typical laboratory setup.

Figure 1 shows the WNI in a laboratory environment, and Figure 2 illustrates a typical laboratory setup, very similar to the one used in the experiments presented in this paper.

The system may be used in lab environment (using the JTAG interface to build the MAC firmware and to download it on the DSP on-board FLASH memory), as for instance during synthetic traffic tests. It may also be used in a real environment, using the high speed IEEE1394a (FireWire) bus, which allows the full speed connection with a host PC running the user applications (not shown in Figure 2).

The RF and BB components have been kept compliant to the 802.11 standard, such as to allow mixed environment experiments, where enhanced systems cooperate with standard, off-the-shelf IEEE 802.11 wireless cards. For the tests, 4 enhanced systems have been assembled.

2.1. Hardware overview

The selection of the hardware was a critical phase in the realization of the WNI, and it required an extensive analysis of components currently available on the market. We had to face different problems in order to be able to find components that could be combined, and to have the required instruments for realizing the 802.11 MAC, maintaining the flexibility needed to support various experiments and allow future extensions.

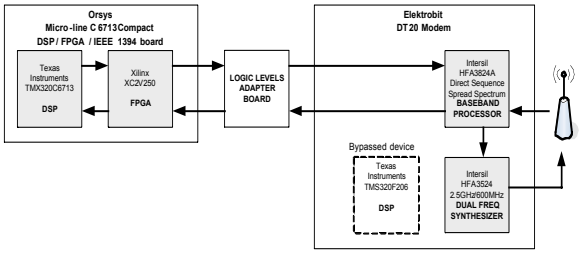


Figure 3: Overview of the enhanced 802.11 Wireless Network Interface (PHY).

The final selection was for the Elektrobit DT20 modem and the Orsys Micro-line C6713Compact DSP board. The Elektrobit DT20 modem performs the BB and the RF processing in compliance with the 802.11 PHY layer using an Intersil Prism-I WLAN chipset, while the Orsys Micro-line C6713Compact DSP board is used for the MAC implementation. RF modem and DSP board are connected through a small custom-made adapter board, which performs a voltage level conversion to adapt the logic signals voltage levels (3.3V for the DSP board and 5V for the modem).

The DSP board integrates a large FPGA (Xilinx XC2V250) and a Texas Instruments TMX320C6713 DSP. The FPGA has been used for interface adaptation between DSP and Intersil components, and could be

used in the future in order to accelerate other tasks (e.g., address filtering, cryptography, etc.). The 802.11 MAC firmware is developed in standard C, but specifically customized for the C6713 DSP. A more detailed overview of the interfaces between the logic components constituting the MAC implementation is depicted in the block diagram shown in Figure 4.

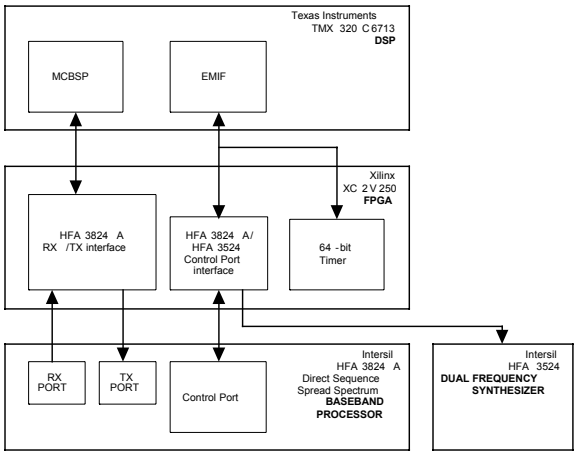


Figure 4: Logic blocks diagram of the MAC implementation.

2.2. Firmware overview

The firmware running on the C6713 DSP is basically a simple monitor loop with few interrupt routines. However, an operating system was not needed for the implementation of the standard 802.11 Frame Exchange Sequence and relative tasks (fragmentation, de-fragmentation, fragment cache control, etc.). Nevertheless, a Real-Time operating system could be easily installed to accommodate more sophisticated procedures. Thus, in the software we have developed, only few components are specific to the C6713Compact board; among these: timing considerations, available DSP resources, configuration and control required for the specific implementation. On the actual system (C6713Compact board), the firmware occupies about 125 Kbytes and can reside completely in the DSP internal RAM, at run time. It is worth pointing out that the source coded we have developed is highly portable because it is not tailored to a specific OS.

Since the original DSP implemented only a basic monitor loop, we had to develop our MAC protocol implementation starting from scratch. We designed our software maintaining the maximum possible abstraction, in order to minimize the software re-design in case of change of the development platform, and to guarantee sufficient flexibility for future extensions. It is worth remarking that there is still a large margin left for the DSP, because the C6713 DSP could easily accommodate more demanding standards such as the 11 Mbps IEEE802.11b.

The firmware we have developed is subdivided into the following components:

- *MAC Firmware*. This is the hard real-time software, which allows packets (fragments) to be physically transmitted to and received from the RF interface.
- *Host Interface Firmware*. This software component managed the communication flows between the host CPU and the DSP. This communications are carried out through the on board IEEE1394a port.
- *Packet Data Management*. To manage the data flow between the communication channel and the host, specialized data structures are needed. Given the hard time constraints of the MAC firmware these data structures must be optimized in terms of numbers of memory access, numbers of data swaps and storage.

The code implementing the 802.11 MAC software is composed of 13 modules (source & header files). Their functionalities and described in [8], but are omitted here for space constraints. Finally, the firmware is linked with the c6x_boardlib, which is a third party library implemented by Orsys for the C6713Compact board.

2.3 Enhanced backoff

In this section we outline the operations of the AOB mechanism [5], which has been used for implementing the enhanced IEEE 802.11 MAC protocol in our WNI. The AOB scheme was proposed to improve the efficiency of the IEEE 802.11 MAC protocol, by extending the standard binary exponential backoff to guarantee that the wireless stations adopt the optimal backoff interval corresponding to the current contention level in the network. The utilization rate of the slots (also denoted as *slot utilization*) is used as an estimate of the current network contention level. The slot utilization can be computed as 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. The slot utilization definition wasn't originally introduced by the AOB mechanism. In [4], the DCC mechanism was proposed that exploits the slot utilization (SU) to decide in a probabilistic way when performing a transmission attempt or further deferring the access to the channel. Specifically, the DCC mechanism computes a *Probability of*

Transmission P_T according to the following formula:

$$P\_T = 1 - SU^{N\_A}, \quad (1)$$

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 perceive the channel as idle), the station will perform a real transmission with probability 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 is capable of limiting the slot utilization value.

Numerical results presented in [4] 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, the DCC solution operates in a heuristic way, using larger contention 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 carried out in [11] and [5] to identify the optimal slot-utilization level the network should obtain to guarantee the maximum channel utilization. Specifically, in [5] it was derived the *Asymptotic Contention Limit* (ACL) function, which approximates the optimal slot utilization with an accuracy that increases as the network contention, i.e., the number of active stations and/or message length, increases. 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:

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

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 analytical characterization of the ACL values presented in [4], demonstrated 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 contention level [2], [6].

The enhanced IEEE 802.11 MAC protocol we have implemented is a slight variation of the original AOB scheme, although it is totally equivalent as far as the protocol behaviour and performance. We decided to do not estimate the aggregate slot utilization, as done in [4], [5], but we split it into two contributions: the *internal* slot utilization (SU_{int}) and the *external* slot utilization (SU_{ext}), such as to differentiate between the contribution to the channel occupation due to the node's transmissions and to its neighbors' transmissions. This is motivated by the need to keep our implementation as much flexible as possible, in order to allow future modifications. In fact, several extensions of the AOB scheme are currently under investigation [12], and the implementation of some of these proposals is an ongoing activity. Another variation with respect to the original AOB is the time interval over which we compute the slot utilization. In fact, the original AOB computes the slot utilization after each backoff interval, while in our implementation we used a constant observation period T equal to 100ms. This choice is motivated by the need to avoid frequent slot utilization computations, which could interfere with the time constraints of the atomic MAC operations (e.g., RTS/CTS exchange). Thus, considering a fixed observation period T, each station computes the internal slot utilization, SU_{int}, and the external slot utilization, SU_{ext}, as:

$$SU_{int} = \frac{N_{tx}}{\frac{T_{idle}}{t_{slot}} + N_{rx} + N_{tx}}, \quad (3.a)$$

$$SU_{ext} = \frac{N_{rx}}{\frac{T_{idle}}{t_{slot}} + N_{rx} + N_{tx}}, \quad (3.b)$$

where N_{tx} is the number of transmissions carried out by the station during the T period; N_{rx} is the number of separate channel occupations (either successful transmission or collisions) observed during the T period; and T_{idle} is time the channel is idle during the T period (including the idle periods during which the $DIFS$ and $EIFS$ timers are active). It is evident that the SU used in Equation (1) and Equation (2) can be computed as the sum of SU_{int} and SU_{ext}. Thus, our implementation and the original AOB are equivalent.

3. Measurement and Results

In order to validate our enhanced architecture we carried out comparative tests of the performance achieved by the legacy IEEE 802.11 backoff mecha-

nism and the enhanced one. In both sets of experiments we used our WNI implementation. All the tests were performed in a laboratory environment, considering ad hoc networks in single-hop configurations. Nodes were communicating in ad hoc mode and the traffic was artificially generated. In our scenarios we used a maximum of 4 stations, due to hardware limitations. However, this is not seen as a problem, because we were already able to demonstrate the performance of our solution and the coherence with previously performed simulations. In Section 3 we further elaborate on this point, providing simulation results to support our claim.

As discussed in Section 2, the average backoff value that maximizes the channel utilization is almost independent of the network configuration (number of competing stations), but depends only on the average packet sizes [11]. Therefore, the *ACL* value for the frames size used in our experiments can be precomputed and loaded in the MAC firmware. The implementation in the FPGA of the algorithm defining the *ACL* value in order to compute it at run-time, is an ongoing activity.

3.1. MAC Performance tests

We used different scenarios (2, 3 and 4 stations), in order to study at the same time, the performance of our implementation and the correspondence with the previous simulation results. The stations were identically programmed to continuously send 500-bytes long MSDUs (MSDU denotes the frame payload). The consecutive MSDU transmissions were separated by at least one backoff interval and we did not use the RTS/CTS handshake, or the fragmentation. The minimum contention window was set to $8 \cdot t_{slot}$ (160 μsec), and all values were computed in stationary conditions. The nodes topology is illustrated in Figure 5. All the experimental results we show henceforth were obtained by computing the average over ten replications of the same test.

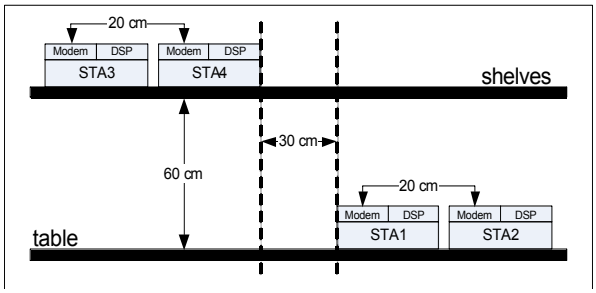


Figure 5: nodes topology used in the measurements.

As already demonstrated in [5] and [12] the AOB mechanism introduces a minimum overhead that could negatively affect the performance of the communications between two stations. Thus, our first set of experiments was carried out to verify this performance decrease in network configurations where two stations are performing either a unidirectional or bidirectional communication, as illustrated in Figure 6.

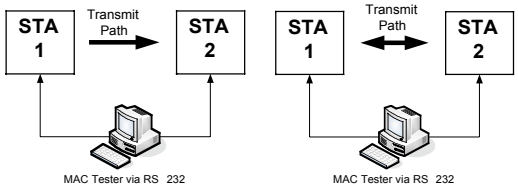


Figure 6: Unidirectional and Bidirectional communications with two stations.

The results we obtained in this point-to-point configuration are reported in Table I. In the table, N_{tx} denotes the average number of transmission (either successes of collisions) performed during a period T, while RC denotes the average number of collisions suffered during a period T. Hence, the average throughout TP, expressed in byte/s, can be computed as:

$$TP = \frac{N_{tx} - RC}{T} \cdot MSDU . \quad (4)$$

From the listed numerical results, we can observe that the throughput decrease in the case of two competing stations is lower than 3%.

Table I: Point-to-point scenario results.

	Unidirectional Flow		Bidirectional Flow	
	Standard MAC	Enhanced MAC	Standard MAC	Enhanced MAC
SUint	0.09645	0.04114	0.05709	0.02407
SUext	0.00024	0.00040	0.05939	0.02312
P_T	-	0.56149	-	0.52652
Ntx	33.60193	29.88714	16.8941	15.99531
RC	0.18143	0.38419	1.16443	0.72829
TP	**167100**	**147550 (-11.7%)**	**78650**	**76350 (-2.94%)**

In the second set of experiments we considered a network configuration with 3 stations, as depicted in Figure 7.

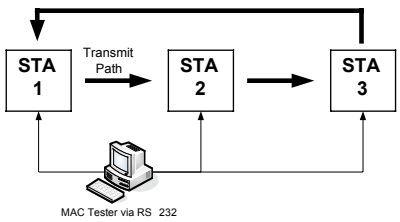


Figure 7: 3 stations scenario.

The experimental results we obtained in the 3 stations configuration are reported in Table II. We can note that with three competing stations, the throughput improvement is about 5.7%. This is explained by observing that the number of collisions occurred during a T period is four times less for the enhanced MAC than for the standard one.

Table II: 3 stations scenario results.

	Standard MAC	Enhanced MAC
SUint	0.04585	0.01911
SUext	0.07509	0.03142
P_T	-	0.49922
Ntx	13.0545	11.4754
RC	2.74377	0.57712
TP	**51555**	**54485 (+5.7%)**

Finally, the last set of experiments was carried out in the 4 stations scenario depicted in Figure 8.

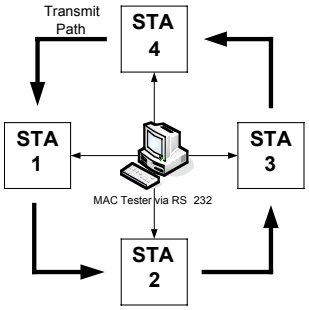


Figure 8: 4 stations scenario.

The experimental results obtained in the 4 stations configurations are reported in Table III. These results confirm the positive trend shown in the previous experiments, since the throughput increase in the case of four stations is 9.4%.

Table III: 4 stations scenario results.

	Standard MAC	Enhanced MAC
SUint	0.03871	0.01517
SUext	0.09371	0.03846
P_T	-	0.44863
Ntx	9.91685	8.8006
RC	2.18539	0.33916
TP	**38655**	**42300 (+9.4%)**

To summarize, in Table IV we report the aggregate throughput measured in the different network configurations we have tested. The numerical results clearly demonstrate that the enhanced MAC protocol may significantly improve the per-station throughput as the number of stations increases.

Table IV: Summary of experimental results: total throughput (Kbyte/s).

	2 STAs (bidir.)	3 STAs	4 STAs
Std MAC	**157300**	**154665**	**154620**
Enh MAC	**152700 (-2.94%)**	**163455 (+5.7%)**	**169200 (+9.4%)**

3.2. Relations with previous simulation results

We carried out experiments with up to four stations due to hardware limitations. However, we argue that the positive trend observed in our tests will be confirmed also for large numbers. To substantiate this statement, in this subsection we show numerical results obtained through discrete-event simulations, considering the same parameter setting adopted during the real tests. In particular, Figure 9 shows the channel utilization of the IEEE 802:11 MAC protocol (rate 2Mbps) with and without the AOB mechanism, versus the number of wireless stations in the network for an ideal wireless channel, i.e., not affected by noise. In the same figure we also show the maximum throughput achieved when the stations adopts the *optimal* backoff interval (computed according to [6]). The shown results refer to a payload size of 500 bytes, and were obtained using a minimum contention window equal to 8 time slots, as in our real experiments.

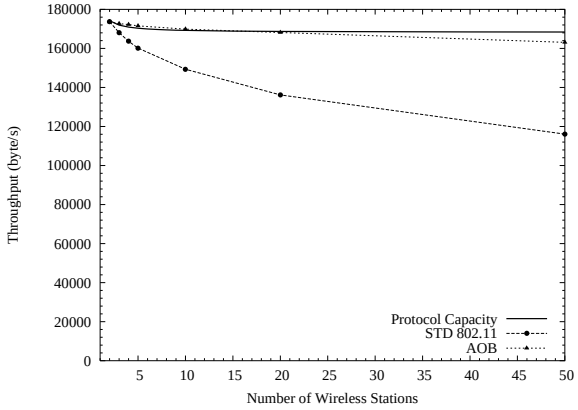


Figure 9: Throughput of the IEEE 802.11 protocol with and without the AOB mechanism versus optimal value.

From the figure, it is straightforward to note that the throughput measured in the real tests is always lower than the one obtained through simulations. The worse throughput performance is due to the negative impact of the radio interferences that are present in a real environment, and this is true for both the standard 802.11 MAC protocol and the enhanced one. However,

the comparison between the experimental results reported in Table IV, and the simulation results shown in Figure 9 indicate the AOB mechanism is quite effective in reducing the throughput degradation caused by the radio interferences. In fact, while the theoretical throughput improvement achieved by the AOB mechanism in a 3 stations scenario is 2.6% in the case of ideal channel conditions, in our tests we measured an improvement of the 5.7%. Similarly, the theoretical throughput improvement achieved by the AOB mechanism in a 4 stations scenario is 4.9%, while in our experiments we obtained a 9.4% improvement. The reason of this better performance is that the efficiency of the AOB scheme increases as the contention in the network increases (as shown in Figure 9), and the frame losses due to channel noise are treated by the MAC protocol as normal collisions. Therefore, we derived another very important conclusion from our experiments: the AOB scheme shows to be useful to reduce the probability of transmitting frames that may get lost due to the errors induced by channel noise, increasing the throughput in a noisy environment.

4. Conclusions and Future Work

In this paper, we presented the experiments we carried out with the implementation of an enhanced IEEE 802.11 MAC card adopting the AOB [5] backoff algorithm. Our wireless card is still fully compatible with current implementations of the IEEE 802.11 technology because the radio part is compliant to the 802.11 standard. However, the experimental results we presented, demonstrate that the enhanced mechanism we have implemented outperforms the standard 802.11 MAC protocol in real scenarios. We have also shown that the advantages of this mechanism go further than the high contention scenarios (e.g., ad hoc networks), for which it was designed, because it is also effective in lessening the negative impact of the external interferences, which traditionally decrease the performances of wireless networks in any environment.

As discussed in the introduction, the wireless network interface is designed for supporting several types of experiments with different enhanced MAC protocols. To this aim, we are working in several multiple directions [13], [14]:

- **Credit-based AOB:** In recent paper [12] the authors show that the feedback-based mechanisms that have been proposed to maximize the IEEE 802.11 MAC protocol efficiency by guaranteeing an optimal time-spreading of the users' access suffer from a significant unfairness problem when employed in heterogeneous wireless networks, i.e., networks formed by both enhanced stations using the modified 802.11 MAC protocol, and legacy stations adopting the standard 802.11 MAC protocol. In particular, any contention control mechanism aiming at enforcing a maximum contention level in the network is vulnerable to the presence of stations that are allowed to exceed this limit. For this reason, we will experiment the implementation of an enhanced version of AOB [12], where there is a component capable of estimating how long the enhanced stations further defer their frame transmissions with respect to the legacy stations, and then exploiting this information to enable the enhanced stations to reclaim new transmission opportunities. Specifically, the enhanced stations earn credits when releasing transmission opportunities granted by the standard protocol. The collected credits are a virtual currency spent by the enhanced stations to reclaim new transmission opportunities.
- **Routing:** In addition to the backoff issue, there is another problem that limits the performances of wireless traffic in WLANs and ad hoc networks: the routing. Indeed, each packet received from 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. In the meantime the SIFS time has gone and the station must run again the backoff mechanisms in order to acquire the channel to forwarding the packet. This adds undesirable delay and overhead at both MAC and routing layer. For this reason we aim at experimenting mechanisms capable of executing routing operations inside the MAC, layer. The solutions we intend to experiment will be based on a next-hop address lookup in conjunction with a path strategy as, for example, the fixed length labels architecture as defined in [7]. Basically, the packet forwarding protocol builds on the IEEE 802.11 DCF MAC protocol by exploiting some additional information delivered in the control packets (RTS/CTS) to allow the forwarding node to determine the next hop node while contending for the channel. Moreover, we intend to add a communication interface between the MAC layer and the routing layer, such as to allow the routing protocol to take its routing decisions exploiting channel information.
- **Cross-layering:** Besides the routing, research has shown that several mechanisms can profit from the knowledge of some parameters that are typically confined at the MAC layer, such as transport, power management, cooperation, etc. Indeed, several network architecture designers foresee the access to MAC parameters for a full integration of mechanisms traditionally working at different layers. This will be enabled by a cross-layering architecture as the one proposed by the MobileMAN project [10]. In this architecture the shared memory component acts as exchange area of networking information (parameters, status, etc.) for all the layers. This allows the MAC layer to distribute

"physical" information up to the higher levels, as well as to profit from some higher layer elaborations too complex to be performed at MAC. A typical example is the interaction between MAC, routing and transport information for congestion and network utilization purposes. If the transport is aware of the links' status, it can distinguish between congestion due to physical failures and congestion due to the amount of traffic, acting consequently. Similarly, the routing can decide different routing paths or strategies, and the MAC can modify the distribution of some information as consequence.

5. References

[1] ANSI/IEEE Std. 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, August 1999.

[2] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(9): 1787-1800, 2000.

[3] Xu Shugong, T. Saadawi. Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?, *IEEE Communications Magazine*, 39(6):130-137, June 2001.

[4] 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*, 60(4):407-430, April 2000.

[5] L. Bononi, M. Conti, E. Gregori. Run-Time Optimization of IEEE 802.11 Wireless LANs performance. *IEEE Trans. Parallel Distrib. Syst.*, 15(1):66-80, January 2004.

[6] F. Calì, M. Conti, E. Gregori. Dynamic Tuning of the IEEE 802.11 Protocol to Achieve a Theoretical Throughput Limit. *IEEE/ACM Trans. Networking*, 8(6):785-799, December 2000.

[7] A. Acharya, A. Misra, S. Bansal. A label-switching packet forwarding architecture for multi-hop wireless LANs, in *Proc. of WoWMoM 2002*.

[8] R. Bernasconi, I. Defilippis, S. Giordano, A. Puiatti. MobileMAN Wireless Network Interface, http://cnd.iit.cnr.it/mobileMAN.

[9] R. Bruno, M. Conti, E. Gregori. Optimization of Efficiency and Energy Consumption in p-Persistent CSMABased Wireless LANs. *IEEE Trans. Mob. Comp.*, 1(1):10-31, March 2002.

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

[11] R. Bruno, M. Conti, E. Gregori. Optimal Capacity of p-Persistent CSMA Protocols. *IEEE Commun. Lett.*, 7(3):139-141, March 2003.

[12] R. Bruno, M. Conti E. Gregori. Distributed Contention Control in Heterogeneous 802.11*b* WLANs, in *Proc. of WONS 2005*, Jan. 2000.

[13] R. Bernasconi, I. Defilippis, A. Puiatti. Issues and implementation plan of an enhanced MAC, draft version available at http://b.die.supsi.ch/projects/MobileMAN

[14] R. Bernasconi, I. Defilippis, S. Giordano. A. Puiatti. an enhanced MAC architecture for multi-hop wireless networks, in *Proc. of PWC2003*, Venice.

Experiments of Ana4: An Implementation of a 2.5 Framework for Deploying Real Multi-hop Ad hoc and Mesh Networks

Nicolas BOULICAULT, Guillaume CHELIUS and Eric FLEURY
CITI – INRIA/ARES
INSA de Lyon – France

Abstract

We consider the problem of interconnecting several hosts in a spontaneous hybrid network, i.e. *an environment where wired and multi-hop wireless technologies are used. Dealing with the issues raised by such hybrid networks and addressing all the challenges listed in MANet may require a departure from classical solutions available in the literature. To enable a full multi-hop connectivity without raising problems/inconsistencies regarding IP compatibility, we have proposed Ana4, a 2.5 layer architecture suitable for hybrid networks. In this article, we present the advantages of Ana4 in the context of ad hoc and mesh networks as well as application fields where Ana4 is already used. We also present experimental measurements that show good performances with respect to a full IP solution (IP routing) and similar 2.5 approaches like MPLS.*

1. Introduction

Wireless communications will play a crucial role in ambient networks that will interconnect a wide range of equipments, from places like homes and offices to public areas. Such ubiquitous computing will indubitably interconnect a set of heterogeneous equipments (personal electronic devices, video entertainment, play station, home appliances) with several heterogeneous physical and link layer technologies, from wired to wireless ones. In such scenarii, wireless devices may be fixed or not and one may imagine that all devices will take profit from a global Internet connectivity. Wireless technologies offer open solutions to provide mobility and services where the installation of a complex wired infrastructure is not possible. Nowadays, performance of wireless local area networks increases with the evolution of the normalized physical layers (802.11b, 802.11a, 802.11g, zigbee, wimax, ...).

Multi hops wireless ad hoc networks and mesh networks offer a promising application field as main actual wireless technologies used by personal devices operate only over short distances. Moreover, it may not be possible to always deploy a networking infrastructure (cellular or wire cable) due to practical or cost constraints. In order to extend the coverage of classical wireless infrastructure-based networks, wireless multi hop networks (or wireless *ad hoc*/*mesh* networks) have been proposed.

In this paper, we present the implementation of a practical architecture suitable for interconnecting devices in an *hybrid network environment*, where wired and multi hop wireless technologies are used. By ad hoc architecture, we denote a set of rules and operations dealing with addressing and routing that must be set up for the ad hoc network to offer basic services such as routing, IP compatibility or Internet connectivity. In particular, such an architecture must answer the two following fundamental questions: what is an ad hoc address? What element is identified by an ad hoc address?

Previous works on ad hoc networks have mainly relied on the IETF MANet working group that proposes an architecture in which the basic element is the network interface and where an ad hoc address is an IP address. Under IPv4, the actual philosophy is to design/implement all MANet routing protocols at the IP level. This architecture is not fully satisfying and introduces several issues/inconsistencies regarding IP

compatibility. As ad hoc packets are routed using IP addresses, there is a *chicken and egg* problem while dealing with IP address auto-configuration in a multi-hop environment: under the MANet philosophy, in order to performe multi hop communications an IP address is mandatory bu it the same time, to obtain an IP address from a DHCP server located several hops away, an host need to perform a multi hop communication. Regarding this architecture, IP broadcasting in an ad hoc network is also a hard task as the IP broadcast address must not be routed/forwarded. Moreover, it is important to notice that, as stated in [4], a MANet node using wireless technologies A and B (*e.g.* frequency A and frequency B) can communicate with any other node possessing an interface with technology A or B. This means that the ad hoc routing must operate over a multi-graph composed of several physical graphs and that an ad hoc node is the union of all its interfaces involved in the ad hoc network. This view increases the routing complexity and the number of control messages when several interfaces of a same ad hoc host are identified by different ad hoc addresses as it is the case in the MANet philosophy. Addressing all the challenges listed in MANet [4] may require a departure from solutions available in the literature. Implementing the ad hoc support below IP is not new. In the past, several architectures have been proposed: LUNAR [12] , ABR [11]. More recently, a solution called *Lilith* for spontaneous networks based on MPLS [13] has been proposed. We will see that all these architectures lack flexibility to be completely satisfying.

In this paper, we present Ana4 [1] an underlay networking mechanism useful for MANETs. Ana4 defines a new generic lightweight and efficient ad hoc architecture, which relies on the notion of ad hoc virtual interface and logical sub-networks. A virtual interface is a logical entity which abstracts a set of network devices into a single addressable network component. In this paper we focus on the architecture and on the implementation and performance measurements of our 2.5 proposal for multi hop hybrid networks. The rest of the paper is organized as follows. We first discuss in section 2 the services that we can legitimately expect in an ad hoc network. In section 3, we present the related works and describe our Ana4 architecture. Then we briefly expose some advanced functionalities of Ana4 in section 4. Section 5 is dedicated to several actual use of Ana4. In section 6 we present some measurement experiments. Finally, we conclude this article with section 7.

2 Plea for an ad hoc architecture

The fundamental service in an hybrid environment is to allow communication between all devices of the network, that is, to able a peer-to-peer mobile routing capability in a purely wireless domain as stated in [4]. Since some nodes may be out of range or since some nodes may not share the same medium (incompatible wireless devices), it is necessary to define complex routing mechanisms that allow multi-hop routing. Inside a given hybrid network, routing mechanisms must be implemented in order to guarantee the intranet connectivity. These mechanisms must ensure unicast and broadcast/multicast routing capabilities. Some other services are also required, such as support for the TCP/IP protocol stack or Internet connectivity. In this section, we give a brief and not exhaustive overview of the main services that one can legitimately expect in an ad hoc or mesh network.

Intranet connectivity. The routing paradigm is the main factor driving the design of all networks. The routing function problem in ad hoc network appears as a crucial point and specific routing algorithms must be derived for ad hoc network. Several researches are done both on proactive and reactive approaches. It is important to notice that, as stated in [4], a MANet node using wireless technologies A and B (e.g. 802.11 and Bluetooth) can communicate with any other node possessing an interface with technology A or B. This means that the unicast routing algorithm must operate on a multi-graph composed of several physical graphs and that an ad hoc node is the union of all its interfaces involved in the ad hoc network. The unicast routing must offer a global connectivity over all the interfaces. The second important service that must be supported in an ad hoc network is the broadcast facility. As in classical networks, a node may need to send a message to all other nodes. This facility is used in almost all unicast routing algorithms [8, 10] developed for MANet networks and must be supported in a efficient way at the ad hoc level.

Complete support for TCP/IP. Once the connectivity is provided, the second service that must offer an ad hoc network is the TCP/IP one as the whole Internet relies on these protocols. Every node must be able to behave as if it belongs to a standard IP network, that is in "*an interoperable inter-networking capability over a heterogeneous networking infrastructure*". Moreover, a partial support or compatibility with IP is an unsatisfying approach. Based on these trivial remarks, we must focus on the consequences they imply. First, IP defines a set of addressing as well as address-related routing rules. For example, it defines the notion of IP networks and IP sub-networks. Routing and accessibility directives are associated to these notions. IP also proposes broadcast notions and rules. For example, a packet directed to the address $255.255.255.255$ is received by all nodes connected to the local link of the source and is not supposed to be forwarded. Numerous protocols and applications rely on IP standards. If we desire a complete compatibility with existing networking environments, the ad hoc architecture must be fully compatible with IP. Secondly, several auto-configuration mechanisms are proposed in association to IP. In IPv4, the DHCP protocol allows an host to retrieve its IP address from a server. If these services are powerful ones in wired networks, their importance is also obvious in ad hoc networks.

Internet Connectivity. In an hybrid context, it is important to offer a global connectivity. The notion of global connectivity is more general than the care of address feature present in IP mobility or the delivery of a global IPv6 address. Offering a global connectivity to the Internet is providing a service continuum. This means for example that an ad hoc node must be able to receive its favorite net-radio multicasted from a server which is not localized within the ad hoc network. As stated above, the multicast protocol used inside the MANet will be specific and the global connectivity service will perform the gateway operations with PIM/CBT/YAM...

Other features. We may want to be able to switch from one physical radio interface to another for power consumption reason for example, without interrupting its sessions or having to deal with any IP mechanism. It means that a micro-mobility (or vertical mobility) support must be implemented inside an ad hoc node so that changing its ad hoc communicating interface will not lead to a change in the IP address used by the host. An other very important feature is the easiness of implementation. Turning a node (PC, PDA...) into an ad hoc node should not require any kernel modification nor any driver modification. Last but not least, the scalability requirement may be an important factor. One can imagine than an ad hoc network may support tens to hundreds of mobile nodes but must also be scalable to a higher factor if this kind of networks becomes really pervasive [7].

3. The Ana4 Architecture

Based on the preceding remarks, it seems important to provide a more fine grain layering that includes an ad hoc level as compared to the pure IP layering. We can locate it at level 2.5, *i.e.*, between level 2 (MAC) and level 3 (IP). Locating the ad hoc level at layer 3 induces several problems with the IP protocol as it is currently known. We argue that in order to be able to deploy ad hoc networks in a very friendly way, a full IP stack support must be provided which also implies to take into account the legacy of already deployed applications/configurations. Likewise, locating the ad hoc level at layer 2 like for the LUNAR [12] architecture induces several issues concerning multiple interfaces support and hardly enables support for micro-mobility. For instance, LUNAR does not address the support of multiple interfaces. Moreover, LUNAR is bound to a specific routing scheme and has been designed for small networks of around 10 to 15 nodes.

Locating the ad hoc layer at level 2.5 is an interesting alternative as it can both enable a full support of IP and provide the opportunity to handle several interfaces. A first step towards this direction was made with an implementation of ABR [11]. However, ABR relies on the IP addressing which leads to the already described auto-configuration issue and, as LUNAR, is bound to a specific routing scheme. Recently, a work posterior to Ana4, Lilith [13] was designed to use MPLS for ad hoc routing. However, this solution is also not satisfying as it does not support multiple interfaces or vertical handoffs, the handling of logical sub-networks or auto-configuration mechanisms and since it is binded to a on purpose proactive like routing pro-

tocol.

3.1 Inter-node architecture

We propose an architecture breaking up an ad hoc network into three levels of abstraction: the hardware level, the ad hoc level and the IP level. If the first one relies on a physical reality, interface communication compatibility, the two others are purely theoritical views. The base element of the first level is the wireless interface whereas for the two others, it is the ad hoc node. Our definition of an ad hoc node follows the one proposed by MANet in [4].

Hardware level. The hardware level is the set of the different hardware networks. A hardware network is the gathering of all interfaces that are physically able to communicate with each others. At this level, the notion of communication ability is related to link layer device compatibility and not to effective communication possibility. In a hardware network, hardware addresses (*e.g.*, MAC) identify interfaces.

Ad hoc level. The ad hoc level defines the ad hoc network. An ad hoc network is the combination of all hardware networks. At this level, the base element is no more the interface but the ad hoc node. We do not make distinction between interfaces but only see nodes with a single interface, the ad hoc virtual interface connected to the ad hoc network. A unique ad hoc device is abstracted from all wireless and wired devices. In an ad hoc network, an element is named using a unique node identifier, also called an ad hoc address. Multi-hop communication is available. One node may send packets to a node distant from several hops. Packets are commuted from ad hoc nodes to ad hoc nodes depending on the ad hoc addresses of destinations. Broadcast and multicast mechanisms are also available. While commuted, a packet may transit through any underlying hardware network and join the destination through any of its hardware interfaces. The followed path is determined by the commutation or routing protocol. This routing protocol is independent of the architecture.

IP level. At this level, an ad hoc network is seen as an Ethernet bus (more precisely as a switched Ethernet link): the IP abstracted network. An ad hoc node of the ad hoc level is looked upon as a single and classical Ethernet interface: the virtual ad hoc interface. In other terms, a node with several physical devices only owns one interface from the IP view. All the commutation work performed at the ad hoc level is transparent to IP.

This architecture allows a complete compatibility with IP. For example, locally broadcasted packets reach all nodes of the ad hoc network without being IP routed. Auto-configuration becomes straight-forward. Retrieving an address at multiple hops through the DHCP protocol is possible since ad hoc nodes are reachable even if not IP configured. As the commutation is performed at the ad hoc level, no IP address is needed to communicate with other ad hoc nodes. More globally, all the IP world behaves as it does with an Ethernet link.

3.2 Intra-node architecture

The role of a virtual ad hoc interface, illustrated in figure 1, is to hide the different physical devices and hardware networks; it provides the illusion of a single virtual network. At the ad hoc level, this virtual network is a wireless multi-hop network; at the IP level, it is a switched Ethernet link. A powerful characteristic of this architecture is to allow an host to use a device simultaneously in ad hoc and in classical modes. Suppose that a physical device handled by a virtual ad hoc interface is also configured as an Internet device. From the IP view, the mobile hosts two distinct interfaces. IP networking is performed over these two interfaces without interference.

The virtual interface. For upper layers, the virtual interface acts as a classical interface. For example, it is declared as an Internet device to the IP layer. IP outputs packets directed to ad hoc nodes through the virtual interface. For the under layer, *i.e.* the link layer, the virtual interface acts as an upper layer protocol. Upon reception of a packet that has transited through the ad hoc network, the packet is given to the virtual interface. This particular architecture is interesting since it does not require any modifications neither in device drivers nor in the TCP/IP stack.

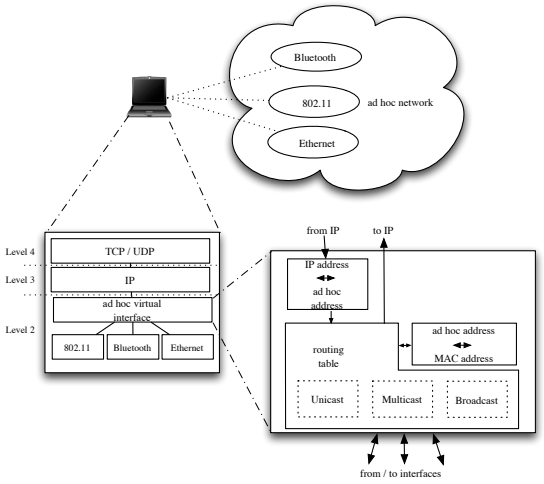


Figure 1. The virtual interface

Naming ad hoc interfaces. Introducing a logical network and logical interfaces require the introduction of a corresponding logical naming process. Virtual interfaces are addressed using ad hoc addresses. An ad hoc address is composed of two fields (Fig 2): the network identifier field, `Net Id`, and the node identifier field, `Node Id`. The last field must ensure the uniqueness of the ad hoc address. It may be configured by hand, chosen using a MAC address or using a statistically unique and cryptographically verifiable (SUCV) identifier as presented in [9]. SUCV identifiers allow authentication of ad hoc nodes in the network.

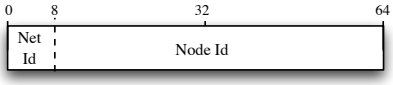


Figure 2. Structure of an ad hoc address

3.3 Commutation

The role of the virtual interface is to commute packets between different devices and upper layer protocols. Upon reception of a packet, the interface decides whether it has to emit the packet, through which interface and to which nodes, and whether it has to forward it to upper layers. A virtual interface owns a commutation table. This table is managed by an application level routing protocol such as the ones studied in the MANet group (OLSR [8], AODV [10] or DSR [6] for example). The routing protocol is independent of the global architecture. Any MANet protocol may potentially be used. Its role is to compute or discover routes and to configure commutation tables in ad hoc interfaces. Very few work is required to adapt a MANet routing algorithm to our architecture. Basically, node identifiers have to be changed from IP to ad hoc addresses.

4 Advanced functionalities

As we have already said, our architecture intrinsically provides a multiple interface support and allows the use of different network protocols over the ad hoc network. Ana4 is an architecture that allows a complete support for IPv4, including auto-configuration mechanisms. It also provides a complete connectivity with the Internet, in regards to routing but also multicast and other services mechanisms. Moreover, Ana4 handles problems related to scalability and network partitioning. Trying to keep a totally flat topology may induce too long flooding delays or huge routing tables. The network identifier field of an ad hoc address allows the setup of sub-ad hoc-networks through the introduction of communication policies between ad hoc nodes with different network identifiers. Ad hoc networks can be split in several logical subnetworks which will appear to IP as different virtual Ethernet links. For more information on Ana4, its functioning or its advanced functionalities, please refer to [2].

5 Deployment of Ana4

Ana4 is currently used in 4 different test beds, two of them in partnership with private companies.

Classical mesh network research test bed. The first “simple” scenario is to use Ana4 to set up a wireless mesh network test bed using an ad hoc routing protocol such as OLSR. By mesh network, we assume a network composed of some fixed and mobile nodes, wireless and wire links and that handles many-to-many connections and is capable of dynamically updating and optimizing these connections. The dynamic management of complex routing information that includes information about external networks (e.g. the whole wide Internet and the gateways to it), is one important key feature inside our research lab mesh network. By

deploying 10 shuttle PCs in the lab, some of them connected to the Internet via a wired link, and by configuring several laptops we are able to use this hybrid network as a regular wireless office covering network. We can come up with relevant real-world scenarios where hybrids between ”static” and ”ad-hoc” networks offer some clear advantages – networks where a number of static nodes form the matrix (the substrate) in which other nodes appear, roam, and disappear. In this mesh network, auto-configuration of mobiles is performed using DHCP.

Heterogeneous seismic sensor network. The IHR [5] project is the second test bed using Ana4. The ambitious goal of the IHR project is to develop a new seismic tool that would allow seismic investigation at scales comprised between one kilometer and few hundreds of meter. The geological targets are those potentially dangerous areas: fault zone, volcanoes, land-slides, valley with site-effect. A new equipment has been recently delivered. The new seismic network consists of thirty nine channels data-loggers equipped with six vertical sensors plus one component sensor. These spider-like mesh networks are connected to each others by network links (wire when possible and/or wireless) that allow a limited crew to control and tune the 270 channels. The deployed seismic network is heterogeneous and scientists on the field encounter several pure networking problems when they try to manage this spider network. How to configure the wireless network, the wire backbone network ? How to deal with IP sub networks... Ana4 offers a simple solution. Due to the Ana4 support for multiple interfaces, the ad hoc network is spread over both the wireless and wire links, hiding the complex and heterogeneous topology to the end user. Auto-configuration is also made straight-forward as the retrieval of an IP address is possible even if nodes are several hops away from the DHCP server. In order to manage the seismic network, scientists also perform broadcast operations in order to start/stop all data loggers. Once more, there is no need to modify all specific application dedicated to the control of the seismic sensors as Ana4 offers a real broadcast to the IP layer even if packets are doing multi hops at the ad hoc layer.

Video Billboard. Those who communicate messages are always in search of attractive ways to carry their information more effectively. When placing outdoor advertising campaigns, 30-second television commercials on large screens (14.69ft x 11.02ft), displaying advertisements, video footage and general information on buildings and walls in busy metro, market, restaurant and/or nightlife districts, the main problem is to carry the messages and update the information. The key idea was to use Ana4 to build a “video billboard mesh network” in order to perform content delivery and network management without deploying a wired network. It’s always more easy to get a power supply on outside building wall than a RJ45 like plug. This project is in partnership with the Embedia company. Embedia is an interactive communications solutions provider. Embedia creates an innovative state-of-the-art link between businesses and their consumers and its patented solution delivers interactive multimedia content directly to end-user devices.

Inventory control and localization. In many industrial contexts (logistics, objects and people monitoring, security...) it may be essential to localize precisely objects or people in real time, whether they are situated indoor or outdoor. In this goal, Kadya is working on providing a localization solution based upon cheap and light emitting radio tags, monitored by a set of listening stations. Each station collects data emitted by the tags through a proprietary radio protocol, and reemits them through a mesh network to a server which analyzes the radio data to calculate the position of each radio tag. Ana4 is used to provide the illusion of a one hop IP network, allowing use of DHCP and broadcast operations for the listening stations. Moreover, the system is easily extensible as any additional station can be placed in the mesh network without the need for any configuration as the multi-hop connectivity is achieved by Ana4 and the IP configuration by DHCP. Furthermore, Ana4 allows the use of IP broadcasting over the network to monitor the system.

6. Performance

We have implemented Ana4 as a kernel module and the code has been released[1]. The first implementation

[1] `http://www.sourceforge.net/projects/ananas`

was done under Linux (PC and PDA) and Windows XP. We provide more details on the implementation in [3] and we summarize below some performance results based on analyses and measures in an experimental network of Ana4 nodes.

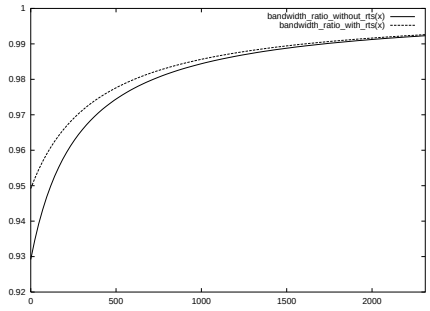


Figure 3. Effective bandwidth ratios on the y-axe as a function of the transmitted packet size (in bytes) on the x-axe.

First, we show the theoretical overhead of Ana4. The Ana4 header introduces an overhead of 160 bits in all packets transiting in the ad hoc network, reducing the performance the network may achieve. However, this overhead is not significant. First, it does not much reduce the volume of data a packet may include. 160 bits is less than 0.8% of a 802.11 link transfer unit (`TU`: 2312 bytes). Second, this overhead does not much lower the useful bandwidth of the medium. It only takes 14us to transfer 160 bits over a link with a 11Mbits/s throughput. In the 802.11 technology, this delay is 8 times smaller than the delay awaited before accessing the medium (`DIFS`$= 128us$). Figure 3 presents some theoretical bandwidth ratios. The medium is a 802.11 link with 11Mbits/s throughput. The x-axis is the size of data in the exchanged packets. The y-axis is the ratio between the effective bandwidths with and without Ana4. The two plots correspond to a classical `RTC-CTS-Data-ACK` transfer scheme and a `Data-ACK` one. In the worst case, packets with only 1 byte of data, the overhead is very low, only 7% loss of the effective bandwidth.

Figure 4. Experimental platform.

To evaluate our Ana4 prototype, we measured its performance in a multi hop network composed of a 4 nodes chain connected via 802.11b wireless or wire links (Fig 4). The goal is to measure the round trip time (`ping`) and the throughput (`netperf`) between nodes at distance 1, 2 and 3 hops (*i.e.*, between nodes: $A-B$, $A-C$ and $A-D$). The comparison was done between: *(i)* a pure static IP route scheme which may be considered as the reference when implementing ad hoc network at layer 3, *(ii)* MPLS commutation with MPLS linux[2] which may also be considered as a reference when implementing an ad hoc solution at layer 2.5 and *(iii)* our Ana4 implementation.

Case	*(i)* static IP	*(ii)* MPLS	*(iii)* Ana4
1 hop: $A-B$	0.099ms	0.114ms	0.115ms
2 hop: $A-C$	0.183ms	0.206ms	0.222ms
3 hop: $A-D$	0.272ms	0.298ms	0.326ms

Table 1. Latency comparison in a wire environment. Average on 100 ping measures with a 64 byte data payload (on top of IP).

Table 1 presents the result of the latency measurements with wire links. The reasons why we used wire links for this measurements is to minimize the medium transmission time in order to isolate the overhead caused by the Ana4 or MPLS layer. We can observe that the overhead introduce by Ana4 is very low. In this overhead, one part is caused by memory operations (introduction and deletion of the Ana4 header). This part can be observed in the one hop measurement. As we can see, this *setup* cost is equivalent for Ana4 (0.115ms) and MPLS (0.114ms) as both protocols perform equivalent operations with their respective headers. The rest of overhead is due to computations (routing table interrogation), at each hops. For this last part, MPLS achieves a better performance (0.092ms/hop for MPLS vs 0.105ms/hop for Ana4) even if the difference is quite negligible (0.013ms). The reason for this is that the MPLS header (32 bits per label pushed in the MPLS label stack) is smaller than the Ana4 one and that the MPLS code is highly profiled and designed for high performances which was not the case with the Ana4 prototype. We are confident on the fact that we could achieve similar results as MPLS with a profiled Ana4 code.

[2] `http://sourceforge.net/projects/mpls-linux/`

Case	*(i)* static IP	*(ii)* MPLS	*(iii)* Ana4
1 hop: $A - B$	5.95Mb/s	5.90Mb/s	5.75Mb/s
2 hop: $A - C$	3.08Mb/s	3.05Mb/s	3.01Mb/s
3 hop: $A - D$	2.07Mb/s	2.05Mb/s	2.03 Mb/s

Table 2. Throughput comparison (`netperf`) in a wireless environment.

Table 2 presents the result of bandwidth performances in a wireless environment. The bandwidth was measured using the netperf software. In the worst case, 1 hop, Ana4 induces a bandwidth lost of 3%. This time, the overhead decreases with the number of hops. The reason for this is that the computation overhead at each hop is recovered by the communications during the transfer. As a consequence, the highest the number of hops, the most negligible it becomes. As for the latency, the difference between MPLS and Ana4 is due to the headers size and the code profiling.

7. Conclusion

We have presented Ana4, a practical architecture suitable for interconnecting devices in *hybrid network environments*. Our goal is to provide a generic lightweight and efficient ad hoc architecture, which relies on the notion of ad hoc virtual interfaces and logical sub-networks. Ananas defines an ad hoc network by introducing three levels of abstraction: the hardware level , the ad hoc level 2.5 and the IP level 3. By designing an ad hoc level at layer 2.5, actual ad hoc routing protocols can be implemented and advanced features like sub-networking, vertical handover or auto-configuration can be offered.

Our first implementation under linux and windows XP shows good performances compared with traditional IP routing and/or MPLS commutation. We observe only a small degradation for the latency and a very small overhead for the throughput when using the Ana4 architecture. However, and as stated before, Ana4 offers advanced features like sub-networking, auto-configuration, vertical handover or all ad hoc node broadcast...

Several possible extensions for this work are open to investigation: taking into account the energy consumption of interfaces to switch from one to another under the virtual ad hoc interface in order to provide the best ratio between energy and throughput for a given class of traffic; auto-organizing the network using the ad hoc sub-network possibilities. Last but not least, we still work on the performance experiments, on the Ana4 code profiling and on the deployment of Ana4 in several scientific or industrial real test beds as mentioned in this paper.

References

[1] G. Chelius and E. Fleury. Ananas : A local area ad hoc network architectural scheme. In *MWCN 2002*, Stockholm, Sweden, Sept 2002. IEEE.

[2] G. Chelius and E. Fleury. Ananas : A New Adhoc Network Architectural Scheme. Research Report RR-4354, INRIA, March 2002.

[3] G. Chelius and E. Fleury. Ananas : A new adhoc network architectural scheme. RR 4354, INRIA, 2002.

[4] S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations. IETF RFC 2501, 1999.

[5] O. Coutant, F. Doré, J. Fels, D. Brunel, M. Dietrich, F. Brenguier, and S. Judenherc. The high resolution seismic imaging (ihr) network, a new tool for seismic investigations at hectometric scales. *Geophysical Research Abstracts*, 7, 2005.

[6] Y. Hu, J. Jetcheva, D. Johnson, and D. Maltz. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet Draft, 2001.

[7] J.-P. Hubaux, T. Gross, J.-Y. Le Boudec, and M. Vetterly. Towards self-organized mobile ad hoc networks : the terminode project. *IEEE Communications Magazine - Special Issue on Telecomuunications Networking at the Start of the 21st Century*, January 2001.

[8] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clausen. Optimized link state routing protocol. Internet Draft), 2001.

[9] G. Montenegro and C. Castelluccia. Statistically Unique and Cryptographically Verifiable Identifiers and Addresses. In *Proceedings of ISOC NDSS02*, San Diego, February 2002.

[10] C. Perkins, E. Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. Internet Draft (work in progress), November 2001.

[11] C.-K. Toh. *Ad Hoc Mobile Wireless Networks*. Prentice hall, 2002.

[12] C. Tschudin, R. Gold, O. Rensfelt, and O. Wibling. Lunar: a lightweight underlay network ad-hoc routing protocol and implementation. In *NEW2AN'04*, St. Petersburg, February 2004.

[13] V. Untz, M. Heusse, F. Rousseau, and A. Duda. On demand label switching for spontaneous edge networks. In *Workshop on Future Directions in Network Architecture (FDNA)*, Oregon, USA, September 2004.

GNU/Linux Implementation of a Position-based Routing Protocol

Marc Heissenbüttel, Torsten Braun, Tobias Roth, Thomas Bernoulli
Institute of Computer Science and Applied Mathematics
University of Bern
3012 Bern, Switzerland
{heissen, braun, roth, bernoull}@iam.unibe.ch

Abstract

The Beacon-Less Routing protocol (BLR) is a position-based routing protocol for mobile ad-hoc networks that makes use of location information to reduce routing overhead. However, unlike other position-based routing protocols, BLR does not require nodes to periodically broadcast hello messages and thus avoids drawbacks such as extensive use of scarce battery-power, interferences with regular data transmission, and performance degradation. In this paper, we describe the implementation of BLR on a GNU/Linux platform comprising laptops equipped with 802.11b WLAN cards and GPS receivers. We present results of BLR's performance obtained from laboratory experiments, which were conducted to validate the implementation for future planned outdoor experiments.

1 Introduction

In position-based routing protocols forwarding decisions are solely based on location information. Each node is aware of its own position, e.g., through GPS, and of its immediate one-hop neighbors by the periodical broadcast of hello messages. Additionally, a location service is required that allows determining the position of the destination node, e.g., GLS [1]. Each node simply forwards a packet to a neighbor which is closer to the destination until the packet eventually arrives at the destination. Many position-based routing protocols have been proposed such as GFG [2], GPSR [3], GOAFR [4]. An overview can be found in [5]. A major drawback of those protocols is the proactive transmission of hello messages which uses scarce network resources such as battery power and bandwidth. Recently, the Beacon-Less Routing protocol BLR was proposed in [6] based on a new routing paradigm enabled by the broadcast property of the wireless propagation medium. Unlike all other routing protocols, forwarding decisions are not taken at the sender of a packet, but in a completely distributed manner at the receivers. A sender does not have to be aware of its neighbors and consequently nodes do not have to proactively transmit hello messages (beacons) as in other position-based protocols, which also saves scarce network resources like battery energy and bandwidth. The performance and the behavior of BLR was studied analytically and by simulations in [6]. Results show that BLR provides efficient and robust routing in highly dynamic ad-hoc networks and is immune to topology changes. Therefore, BLR is especially suited for vehicular and sensor networks with frequently changing topologies. The promising results are the motivation to go a step further and implement the protocol in a real testbed. We developed a prototype system and conducted measurements to obtain more insight on the protocol's performance and behavior in a real world environment. In Section 2, we briefly review the BLR protocol and describe its main features. Afterwards, the implementation on a GNU/Linux platform is presented in Section 3 and encountered real world challenges are discussed in Section 4. Section 5 describes the experimental setup and provides measurement results. Finally, Section 6 concludes the paper.

2 Beacon-Less Routing Protocol (BLR)

Unlike other position-based routing protocols, BLR does not require the periodic broadcast of hello packets. BLR selects a forwarding node in a distributed manner among all its neighboring nodes without knowing the existence or positions of neighbor nodes. BLR has three main modes of operation; greedy mode, backup mode, and unicast mode.

2.1 Greedy mode

Packets are routed in greedy mode whenever possible because only in this mode BLR is really stateless and does not require the transmission of hello packets, i.e., nodes are normally not aware of any neighboring nodes. Therefore, when a node has to send a packet it simply broadcasts the packet.

Consequently, all neighbors receive the broadcast packet. The protocol ensures that just one of the receiving nodes relays the packet further. This is accomplished by different forwarding delays and restricting the nodes that are allowed to forward the packet to a certain area, called forwarding area. Nodes within this area can mutually receive each others transmissions. For the forwarding area BLR uses a circle with diameter r relative to the forwarding node S in the direction of the final destination D as depicted in Fig. 1. A

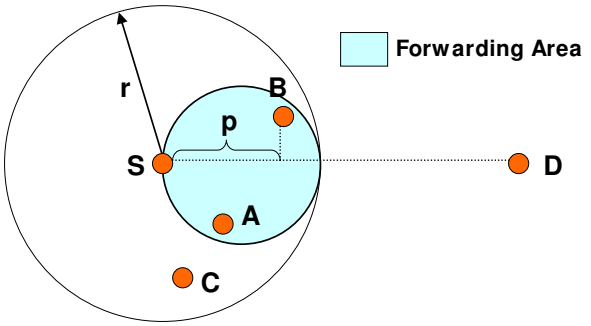


Figure 1. Forwarding Area with potential forwarders A and B

receiving node can determine if it is within the forwarding area from its own position and the positions of the destination D and the previous node S. Both positions of S and D are stored in the packet header. Nodes in the forwarding area are called potential forwarders, e.g., A and B in Fig 1. Potential forwarders calculate a Dynamic Forwarding Delay (DFD) in the interval $[0, Max\_Delay]$ depending on their position relative to the previous and the destination node. The DFD is calculated by (1) with r as the transmission radius of a node, p the node's progress towards the destination, and Max_Delay as a system parameter. Nodes outside the forwarding area simply drop the packet (node C).

$$Add\_delay = Max\_Delay \cdot \left(\frac{r-p}{r}\right) \qquad (1)$$

According to this DFD function, the node with the most progress (e.g., node B), i.e., closest to the destination, calculates the shortest Add_Delay and thus rebroadcasts the packet first. The other potential forwarders (e.g., node A) overhear this further relaying and cancel their scheduled transmissions of the same packet. The rebroadcast packet is also received by the previous transmitting node and acknowledges the successful reception at another node. Simultaneously, the neighbors of the rebroadcasting nodes also received the packet and they determine if they are within the forwarding area relative to node B and destination D. Potential forwarders calculate an Add_Delay and compete to rebroadcast the packet again.

2.2 Backup Mode

If no node is located within the forwarding area, greedy routing fails. This is detected if a node does not overhear a further rebroadcast within Max_Delay of its previously broadcasted packet. This node forwards the packet via unicast further in backup mode. Therefore, the node broadcasts a request for a beacon packet. All neighbors that receive this packet reply with a beacon indicating their positions. The packet is then forwarded to the replying node that is closest to the destination. If none of the neighbors is closer to the destination than the requesting node, the packet is routed according to the face routing algorithm based on the "right-hand" rule, a concept known for traversing mazes, on the faces of a locally extracted planar subgraph, see for example GOAFR [4] for more details. As soon as the packet arrives at a node closer to the destination than where it entered backup mode, the packet switches back to greedy mode.

2.3 Unicast Mode

Routing in greedy mode makes BLR susceptible to packet duplication as data packets are broadcast over multiple hops. Packet duplication occurs for each node in the forwarding area, which does not detect that a packet was already rebroadcast. In reality, there are many reasons that prevent nodes from successfully receiving the rebroadcast packets such as irregular transmission ranges, obstacles, and simultaneously on-going transmissions in the vicinity. BLR implements the unicast mode to minimize the number of duplicated packets. After a node has detected that another node has rebroadcast the packet, it is also aware of the forwarding node's position. Thus, the node may send the subsequent packets to the same destination via unicast to the node which relayed the broadcast packet. Due to the mobility of the nodes, nodes located at a better position may enter into the node's transmission range. In order to be able to detect these new nodes, a packet is broadcast in greedy mode after a certain time again such that potential forwarders compete to rebroadcast the packet.

3 Implementation

3.1 Overview

The target platform of the implementation is GNU/Linux. We used Gentoo Linux [7], although any other GNU/Linux distribution based on Linux 2.6 will work for our implementation. We integrated BLR within the protocol stack as depicted in Fig. 2, i.e., between the IP and the link layer. Therefore, it is transparent to the upper layers and applications. Consequently any application such as HTTP, ssh, ping and also ICMP can run unmodified.

The BLR protocol was however implemented in the user space of Linux due to simplicity reasons. Therefore, outgoing packets (solid line) have have to be intercepted and processed accordingly before being passed to the wireless network adapter. More specifically, we introduced a virtual interface `tun0` provided by the `tuntap` [8] device. A new route that redirects all traffic to the BLR network (private destination IP-Addresses 10.0.1.0/24) through `tun0` is added to the system routing table. Consequently, Internet traffic is not affected by the BLR application and routed as normal directly to the 802.11 interface. By listening on `tun0`, the BLR application can catch all traffic sent to the BLR network and inserts the BLR header and updates the IP header. Afterwards, packets are sent via the `pf_packet` facility, which allows the sending of Ethernet and IP packets directly to the 802.11 network adapter. Incoming packets (dashed line) are passed

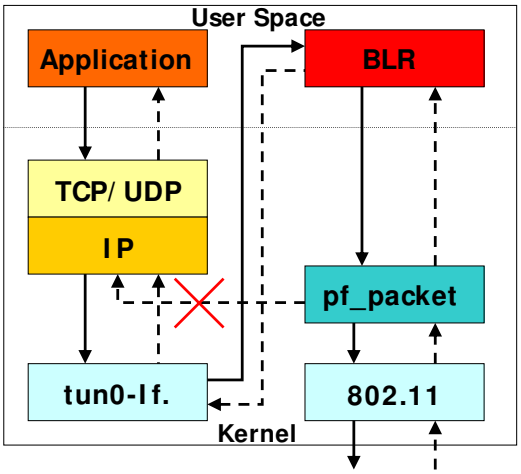


Figure 2. Implementation of BLR in the protocol stack

over `pf_packet` to the BLR application and are either forwarded to the next hop or passed to localhost, depending on the destination address in the IP header. When packets are forwarded, the BLR application only updates the BLR header and additionally delays the packets by the newly calculated Add_Delay before the packets are passed again via `pf_packet` to the network adapter. On the other hand, when the packet is destined for this host, the BLR header is stripped off and the IP header modifications done by the BLR application at the sender are reversed. Afterwards, the packet is forwarded through the `tun0` to the application.

A problem occurs because `pf_packet` actually creates a copy of all incoming packets. One copy is passed to the BLR application, while the original packet is passed to the kernel and from there to the application. The original packet has to be blocked somehow. This is achieved by deploying the IPtables [9] packet filter right after the `pf_packet` facility. This filter blocks all incoming traffic that has the protocol number of BLR set in the IP header. For broadcast packets, this blocking would not be necessary since the kernel simply drops broadcast traffic with a protocol number for which there is no open socket. However, when the kernel receives unicast traffic with an unknown protocol number, it sends an ICMP destination unreachable message back to the sender, which has to be avoided.

3.2 BLR Application

The BLR application is split into three separate processes as depicted in Fig. 3. The main process receives/sends the packet from/to the localhost/network, transforms and updates headers, calculates the Add_Delay, and manages packet timeouts, unicast route information, as well as a list of duplicate packet IDs. The GPS process is connected to an external GPS device and provides location information. The sendqueue process receives outgoing packets together with the calculated Add_Delay from the main process and sends the packet after the indicated delay. If the main processes receives a packet from `pf_packet`, it calls the sendqueue process in order to determine if the packet is queued for transmission. If so, another node forwarded the packet first and the sendqueue process can remove the packet from the queue.

The size of the BLR header is 32 bytes and has the following fields.

- Packet type (1 byte): Data, Location request, Location-Reply, Request for beacon, Beacon.
- Original protocol (1 byte): Protocol number of TCP, UDP, ICMP, etc. This corresponds to the protocol field in the IP header, because the BLR header is inserted between the IP and transport layer header.
- Sequence number (2 bytes): This number together with the source address allows to unambiguously identify a packet.
- Backup distance (4 bytes): This field is used to indicate the distance to the destination from where greedy routing failed, which is required in order to determine when to switch back to greedy routing.
- Position information (8 bytes each): Position of the previous, source, and destination node. The previous and destination node positions are required to calculate Add_Delay. The source node's position is used to update location information at the destination in case of bidirectional traffic.

In the following, we describe in more detail the processes and how packets are handled.

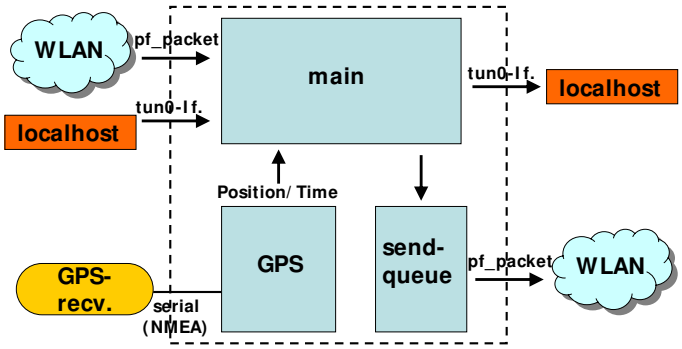


Figure 3. Running processes in the BLR application

GPS Process

The GPS process is connected to an external GPS device, which it polls for changes in location information. It parses the GPS data and passes position updates to the main process. The connection is established through an RS-232 interface and the GPS information is transferred with the NMEA-0183 protocol [10] from the GPS receiver to the laptop.

Sendqueue Process

The sendqueue process is responsible for queuing the packets according to their respective dynamic forwarding delay. It receives packet/delay tuples from the main process and maintains an ordered list of all pending packets. When the associated timer expires, the packet will be sent to `pf_packet`. The sendqueue is implemented separately from the main process since it its only tasks are to manage packet delays and to queue and send packets. The sendqueue further handles the deletion of packets from the queue, whenever the main process detects that another node has already forwarded a pending packet.

Main process

This is by far the most complex process and is responsible for switching packets between components, management and coordination of the other components, execution of the BLR functions, etc. When the main process receives an IP packet through `tun0`, it inserts the BLR header and updates the IP header. Changes are necessary in four IP header fields.

- The source address needs to be changed from the IP address of the `tun0` interface to the IP address of the outgoing interface.
- The packet length fields needs to be increased by the size of the BLR header.
- The protocol field is changed to 254, which we used for the BLR protocol. The original protocol number is stored in the BLR header.
- Finally, the header checksum needs to be recalculated.

Furthermore, the main process calculates Add_Delay and forwards this along with the packet to the sendqueue process. It also maintains a `hosttable` to store information about known destinations, namely their most recent positions and the next hop to reach them if unicast mode is used. The `packetlist` caches packets that have been sent and have not yet been acknowledged, together with a timeout value for each packet. `packetlist` also handles retransmissions in case of timeouts. Whenever BLR has to switch to backup mode, it takes some time until the next hop is determined due to the sending of the beacon request packet and the time until the beacons from the neighbors are received. The `backupqueue` caches outgoing packets that have to wait to be forwarded until the backup mode setup is completed.

4 Challenges

In this section, we briefly review the main challenges we faced during the implementation in the Linux testbed as opposed to the previous implementation in the simulator.

4.1 Location Service

Location services that provide the position of nodes is a research aspect in itself and several solutions have already been proposed (see [5] for an overview). Therefore, a common assumption of most position-based routing protocols is that the position of the destination is somehow known. In the network simulator, it can be implicitly assumed that this position information is available. In reality, we have to implement a mechanism that provides the position. As it was not our goal to implement a fully functional location service and it would not be appropriate for a small testbed with a few laptops, we chose to implement a simple request-reply mechanism based on flooding. The request and the reply with the geographical position are piggybacked on data traffic whenever possible. In case of unidirectional traffic, position information is invalidated periodically, and the source broadcasts a new location request. In case of bidirectional traffic, or simply if TCP is used, destination locations are not invalidated, but the position can be simply extracted from packets returning from the destination, namely from the source field in the BLR header. Thus, the overhead can be reduced to the initial flooding of one location request packet in case of bidirectional traffic.

4.2 Duplicated Packets

The objective of the unicast mode is to reduce the number of duplicated packets. However, still transmissions are broadcast over intermediate hops. In ideal conditions of a network simulator, radio propagation is modeled by simple isotropic transmission ranges. In reality however, we observed many duplicated packets due to irregular transmission ranges. Therefore, we additionally implemented a filtering mechanism. Each node compares the uniquely identifying source address and sequence number of a packet against a table containing the recently received and also overheard packets. If this packet is a duplicate of a previously received or overheard packet, the node broadcasts a control packet suppressing the further forwarding of that duplicated packet by its neighbors. Therefore, the duplicated packet can be again disposed.

4.3 IP Fragmentation

The BLR header is part of the IP payload in the current implementation. Thus, if fragmentation occurs, only the first IP fragment will contain the BLR header. The header is however required to route packets by BLR. Subsequent fragments will not contain the BLR header and will simply be dropped, because nodes do not know how to process them. Therefore, IP fragmentation has to be avoided. To achieve this, the MTU of the virtual tunnel interface is decreased by the size of the BLR header, which is inserted before Transport layer header, in order to avoid fragmentation at the source node. Additionally the DF (Don't Fragment) bit is set in the IP header such that intermediate nodes do not fragment the packet. PMTU (Path MTU) discovery is used to handle links where the standard MTU is too large.

4.4 MAC layer control

If a unicast packet is not acknowledged, the 802.11 MAC layer retransmits a packet up to seven times before giving up. In the network simulator implementation, the MAC layer can signal a failed transmission to the upper layer, which in turn selects another next hop and passes the packet again to the MAC layer. This mechanism is also applied by BLR [6] and GPSR [3]. Without this optimization, many unicast packets would be dropped due to unreachable neighbors, and recovery is left to TCP or the application. This severely decreases network performance as retransmissions are end-to-end and not link retransmissions. In a Linux implementation of BLR with WLAN cards however, the MAC protocol is largely implemented in the firmware of the 802.11b card, which makes accessing the mentioned functions in today's card nearly impossible.

4.5 Interrupt granularity

The Max_Delay can be chosen in the order of some milliseconds based on the experienced network simulator results. Basically, Max_Delay indicates the range over which potential forwarders schedule their retransmissions. The Linux kernel has a limitation that severely affects the possible value of Max_Delay, namely the granularity of the timer interrupts. This granularity is defined by a compile-time kernel constant called `HZ`. On Linux kernels 2.6 or newer, this constant is set to 1000 resulting in timer interrupts every 1 millisecond. (In kernel 2.4 and older, the `HZ` was set to 100). This means that the `select()` system call returns at 1 millisecond intervals only. Consequently the granularity of Add_Delay is also only 1 millisecond. Therefore, a rather long Max_Delay has to be chosen to reduce the risk that all nodes transmit simultaneously and limit the usefulness of the DFD concept. In [6], it was proposed to set $Max\_Delay = 2\,ms$ based on simulation results, which is definitely too short for the Linux implementation. However, the longer Max_Delay also increases the end-to-end delay. While possible in theory, a further increase of the `HZ` value is not yet completely supported by the Linux kernel. Even if possible, increasing `HZ` also increases the overall timer overhead, because more timer interrupts are generated. This may not be an issue for our testbed where no other applications are running, but definitely it will be an issue for small mobile devices with limited computation resources.

5 Experiments

5.1 Equipment and Configuration

The testbed consists of 5 laptop computers running Linux 2.6. Each laptop is equipped with an IEEE 802.11b WLAN cards. The cards are configured to run in ad-hoc mode without RTS/CTS, i.e., the DCF of 802.11b is used, and the data rate is set to 2 Mbps. The hardware equipment is heterogenous, i.e., the laptops are from different manufacturers. The same applies to the WLAN cards, some laptops have built-in cards, while other use Orinoco WLAN cards plugged in the PCMCIA-slot. Each laptop also has a GPS receiver connected via the serial RS-232 line. The GPS devices are not only used for providing positioning information to the nodes, but we also use GPS for timing information. This information is provided once per second. The GPS timing information is actually not required for the BLR protocol, but only for performance measurements. The accuracy of the information is below $5\,m$ and $200\,ns$ for the positioning and timing information, respectively.

In this paper, we present the results from experiments that were conducted in the laboratory in order to validate

the implementation and for reference purposes, which allow a comparison with future outdoor experimental results. As the GPS receivers do not work in indoor environments, the position of the laptops had to be hardcoded to yield a virtual topology. Therefore, the positions and the distances between nodes in this virtual topology do not match the actual physical location of the laptops. Furthermore, all laptops are placed on a table within a few meters of each other and thus could physically receive all transmissions of all nodes. To ensure that a laptop only processes the packets from laptops within the transmission range in the virtual topology, a filter based on MAC addresses has been implemented. This filter operates directly on the `pf_packet` socket and simply drops packets from out-of-range nodes in order to match the physical and the virtual topology such that the BLR application never sees packets from virtually out of range nodes. This approach saves processing work on the side of the BLR application since the kernel does all the necessary filtering. The implementation of the MAC filter is done by means of the Berkely Packet Filter (BPF) language [11]. The GNU/Linux implementation is called Linux Socket Filter (LSF) and is compatible with the BPF language.

Traffic is sent by the ping utility, which yields the round trip time RTT. For each measurement, 2000 ICMP echo requests were sent, which together with the echo replies result in 4000 total data packets. The transmission rate had to be limited to 10 echo request per second, because all nodes are physically within each other transmission range. A transmission of a node blocks all other nodes on the MAC layer, and not only the neighbors in the virtual topology. Therefore, experiments with higher data rates where a new IMCP echo request is sent out before the previous echo reply arrived back at the source do not make sense. The default packet size was set to 56 bytes. Including the ICMP, IP, BLR, and MAC header this yields 180 transmitted bytes. The experiments were conducted with a rather long Max_Delay of $5\,ms$ and $25\,ms$ to reduce the risk that nodes transmit simultaneously due to the low interrupt granularity as explained before in Section 4.5. The transmission range for calculating the Add_Delay was set to a $250\,m$. Except for one experiment, we did not use the unicast mode in order to route packets as often as possible in greedy mode. We used four topologies for the laboratory experiments as depicted in Fig. 4, called chain, pairs, contention, and backup topology. Additionally, we also compared the measurements of these experiments with results obtained from simulations conducted with the Qualnet [12] network simulator and from analytical prediction. The scenarios for the simulations were identical to the experiments, specifically the network topologies and the parameters of the BLR protocol such as Max_Delay.

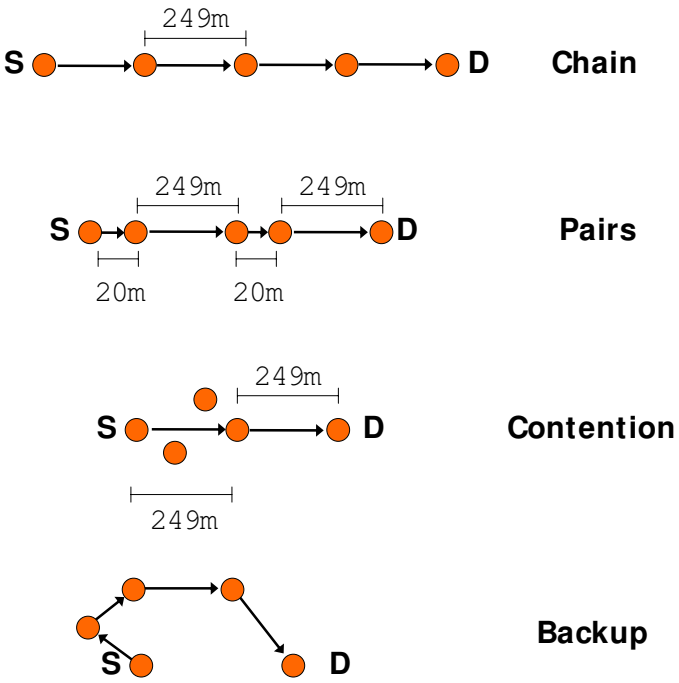


Figure 4. Topologies for the experiments

5.2 Chain Topology

The chain topology is the most simple topology as no contention occurs and only one node always will forward the packet. The forwarding node is located at the boundary of the transmission range and almost immediately forwards the packet without introducing Add_Delay. Thus, the RTT is basically independent of the Max_Delay as shown in the histogram in Fig. 5, which shows the distribution of the measured RTTs. The average is in both cases $17.4\,ms$ and the delivery ratio was always 100%. Considering the fact that a packet pair is transmitted over eight hops (four hops from the source to the destination for the echo request and four hops back to the source for the echo reply), the measured RTT is approximately only $2\,ms$ per hop. When we roughly estimate that 180 Bytes are transmitted over 8 hops with a bandwidth of 2 Mbps, we would expect an RTT of approximately $6\,ms$. The Add_Delay does only contribute marginally to the RTT and is much less than $1\,ms$ per hop, because the progress of $249\,m$ almost equals the transmission radius. In the simulations, we measured an RTT of approximately $8\,ms$, which is close to the analytical estimation, considering that we did not take into account the influence of the MAC layer. However, the RTT is only about half the RTT measured in the experiments. The reason is that the Qualnet network simulator does not introduce any delay for processing packets at the nodes, i.e., the

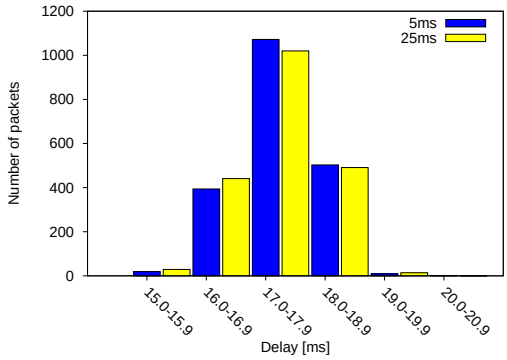


Figure 5. Chain topology with $\mathbf{Max\_Delay = 5\,ms}$ and $\mathbf{Max\_Delay = 25\,ms}$

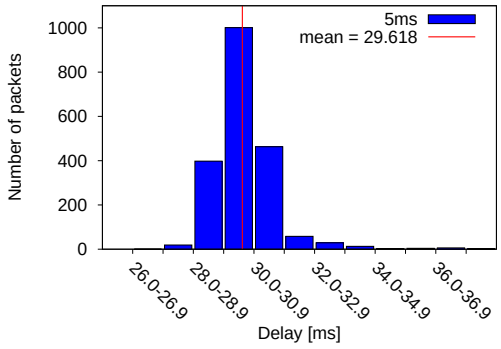


Figure 6. Pairs topology with $\mathbf{Max\_Delay = 5\,ms}$

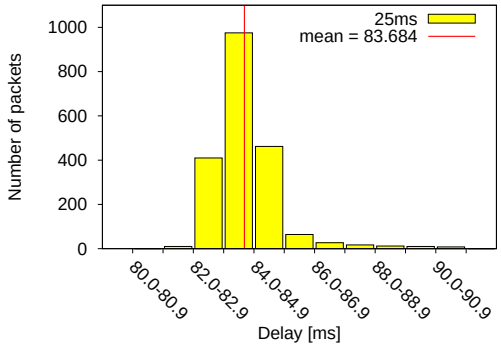


Figure 7. Pairs topology with $\mathbf{Max\_Delay = 25\,ms}$

packets can be forwarded immediately, which is definitely not the case in reality. Furthermore as mentioned in Section 4.5, the Add_Delay has only a granularity of $1\,ms$ in the experiments. This is unlike for the simulations were the Add_Delay is really the calculated value and not "rounded up" to the next millisecond.

5.3 Pairs Topology

The results from the pairs topology are given in Fig. 6 and Fig. 7. In this topology, a packet pair is again transmitted over eight hops. However, the RTTs now vary strongly for the two different Max_Delay as expected. The two transmissions from the nodes with only $20\,m$ progress are delayed significantly as they calculate a long Add_Delay, which is close to Max_Delay according to (1). In the pairs topology, this yields RTTs of approximately $30\,ms$ and $80\,ms$ for $Max\_Delay = 5\,ms$ and $Max\_Delay = 25\,ms$, respectively. The delay introduced by BLR is approximately three times $\frac{230}{250} \cdot Max\_Delay$, two times from S to D and only once from S back to D, which is approximately $14\,ms$ and $69\,ms$ for a Max_Delay of $5\,ms$ and $25\,ms$, respectively. Together with the transmission delay of $6\,ms$, we obtain an expected RTT of $20\,ms$ and $75\,ms$ for the two different Max_Delay values. The respective measured RTTs were $21\,ms$ and $77\,ms$ in the simulations. These results are again approximately $8\,ms$ shorter than in the experiments, independent of the Max_Delay, which confirms the previously stated reasons for the longer delay, namely the zero processing time at the nodes in the simulator and the interrupt granularity of the Linux implementation.

In the pairs topology, we also evaluated the impact of the unicast mode. Although packet duplication is not an issue as only one potential forwarder exists, the RTT is affected. Recall that in unicast mode, the packets are forwarded without introducing Add_Delay if the next hop is known. In Fig. 8, the histogram of the measured RTTs with $Max\_Delay = 5\,ms$ is shown. The RTT is significantly shorter than when packets are always broadcast in greedy mode and is reduced from $29\,ms$ to $16\,ms$. We can also see that there are some packet with longer RTTs around $25\,ms$. The reason is that the unicast mode switches to greedy mode every $5\,s$ in order to detect possibly better located neighbors. Packets transmitted in greedy mode are again dynamically delayed at each node and not immediately forwarded as in unicast mode.

5.4 Contention Topology

In the contention topology, three nodes receive the transmitted packet from the source node and schedule the packet for forwarding as they are all within the forward-

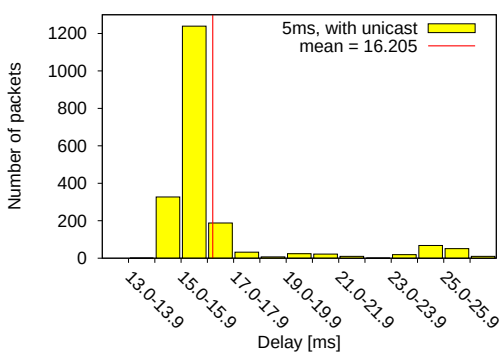


Figure 8. Pairs topology with unicast mode and $\mathbf{Max\_Delay = 5\,ms}$

ing area. They calculate different Add_Delay however and the first transmitting node suppresses the others accordingly. In Fig. 9, the distribution of the RTTs is shown for a Max_Delay of $5\,ms$. The results were almost identical for $Max\_Delay = 25\,ms$ due to the same reasons as for the chain topology. We can observe that the RTT is quite short compared to the previous investigated topologies because a packet pair is only transmitted over four hops (two hops to the destination and two hops back to the source). The theoretical transmission delay is approximately $3\,ms$ because of the reduced hop count. Since the Add_Delay is again for all nodes significantly below $1\,ms$, the expected RTT is around $3\,ms$, which matches the measured RTT of $4\,ms$ in the simulations. The difference to the RTT of the experiments is again because of the required processing time at the laptops.

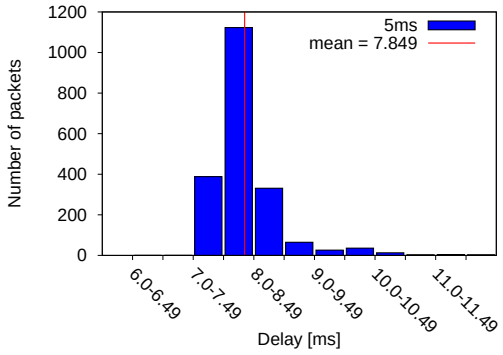


Figure 9. Contention topology with $\mathbf{Max\_Delay = 5\,ms}$

5.5 Backup Topology

In a last experiment, we validated the backup mode of BLR. There is no node located in the forwarding area and the packets are routed in backup mode for three hops until arriving at the node closer to the destination than the source. We measured two different RTTs of approximately $40\,ms$ and $60\,ms$ as depicted in Fig. 10. The reason is that while the backup mode acquires neighbor information if greedy forwarding failed, i.e., during the beacon request reply dialog, other arriving packets are queued in the `backupqueue`. When the backup mode setup is completed and the forwarding node has determined the next hop by the "right-hand", all queued packets are sent immediately to this next hop, thus, some packets in the queue encounter shorter RTTs. The backup mode is also stateless and does not store positions of neighboring nodes, therefore the first following packet after the queue has been emptied again has to wait until the request reply dialog is completed in order to acquire the positions of the neighboring nodes. The RTT is still quite short considering the fact that nodes have to transmit a request for beacon packet and wait until neighbors have replied. In the simulations, we measured an RTT of $51\,ms$ which is again approximately $1\,ms$ less delay per hop than in the experiments due to the same reasons as mentioned before.

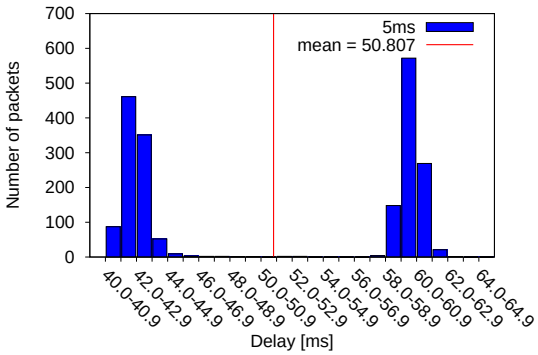


Figure 10. Backup topology with $\mathbf{Max\_Delay = 5\,ms}$

5.6 General Observations

Until now, we only considered the measured RTTs in the experiments. We can conclude that the RTTs are short, are as expected, and vary only slightly around the mean. Other quantitative performance measurements are briefly discussed in the following. In all four topologies, the delivery ratio was always 100%. This is not surprising consider-

ing the fact that the nodes are physically close to each other, even if they are distant in the virtual topology. Furthermore, we observed that in the chain, pairs, and contention topology, there was approximately one packet per thousand packets transmitted unexpectedly in backup mode. The reason was that very rarely some packets showed a higher RTT and collided with subsequent transmitted packets which caused the required retransmissions in backup mode. This effect is especially obvious in the laboratory where all nodes are within transmission range. Furthermore, we observed very infrequently duplicated packets, again in the order of some few per thousand. However, they could be successfully suppressed at the next node by transmitting a control packet as described in Section 4.2. Thus, no duplicated packet arrived at the destination node. Especially for the contention topology, the few duplicated packets indicate that the first transmitting node is able to successfully suppress the other potential forwarders.

6 Conclusions

In this paper, we presented an implementation of the position-based routing BLR on a GNU/Linux platform using laptops equipped with 802.11b WLAN cards and GPS receivers. The advantages of BLR are that it is stateless and does not require to have knowledge about its neighbors, which allows the disposal of the periodical transmission of hello messages and makes it immune to frequently changing network topologies. The BLR was implemented in the user space of GNU/Linux and is transparently integrated in the protocol stack, which allows to run arbitrary applications without modification. We discussed several problems encountered during the implementation and the experiments and described possible ways to solve them. BLR is implemented to retrieve position information provided by GPS receivers. Unfortunately, this information could not be used in our laboratory experiments as all laptops were within a single transmission range and a virtual topology had to be configured manually. We conducted several laboratory experiments to validate the implementation. The forwarding of the packets in the greedy, unicast, and backup mode of BLR was as expected. The results also indicate that BLR is able to deliver packet over multiple hops in a short time. Packets are forwarded reliably and the delivery ratio was always 100%. Furthermore, the forwarding nodes successfully suppressed the other potential forwarders and acknowledge also the previous node reliably, because basically no duplicated packets were observed. In a next step, we will conduct outdoor experiments and use GPS position information. In these experiments, the results may differ from the laboratory experiments because nodes may no longer be within transmission range, which can cause duplicated packets and longer RTTs. We also plan to conduct experiments with high mobility, for which BLR was originally designed, where the laptops are transported in cars.

References

[1] J. L. et al., "A scalable location service for geographic ad-hoc routing," in *Proc. of MOBICOM '00*, Boston, USA, Aug. 2000, pp. 120–130.

[2] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *Proc. of DIALM '99*, Seattle, USA, Aug. 1999, pp. 48 – 55.

[3] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. of MOBICOM '00*, Boston, USA, Aug. 2000, pp. 243–254.

[4] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing," in *Proc. of MobiHoc '03*, Annapolis, Maryland, USA, June 2003, pp. 267 – 278.

[5] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad-hoc networks," *IEEE Network*, vol. 15, no. 6, pp. 30–39, Nov. 2001.

[6] M. Heissenbüttel, T. Braun, T. Bernoulli, and M. Wälchli, "BLR: Beacon-less routing algorithm for mobile ad-hoc networks," *Elsevier's Computer Communications Journal (Special Issue)*, vol. 27, no. 11, pp. 1076–1086, July 2004.

[7] (2005, Apr.) Gentoo linux website. Gentoo Foundation, Inc. [Online]. Available: http://www.gentoo.org

[8] (2005, Apr.) Virtual point-to-point(tun) devices. Maxim Krasnyansky. [Online]. Available: http://vtun.sourceforge.net/tun/index.html

[9] (2005, Apr.) The netfilter website. Harald Welte. [Online]. Available: http://www.netfilter.org

[10] (2002, Jan.) National Marine Electronics Association (NMEA). [Online]. Available: http://www.nmea.org/pub/0183/

[11] S. McCanne and V. Jacobson, "The BSD packet filter: a new architecture for user-level packet capture," in *Proceedings of the 1993 winter USENIX conference*, San Diego, CA, USA, Jan. 1993, pp. 259–269.

[12] (2004, Nov.) Qualnet. Scalable Network Technologies (SNT). [Online]. Available: http://www.qualnet.com/

Implementation Strategies for a Secure and Efficient Multi-hop MANET Platform

Minmin Tu, Jingyu Zhou, Guozhi Xu
Department of Electronics Engineering
Shanghai Jiaotong University
Huashan Road 1954
Shanghai 200030, P.R.C
{tuminmin,jyzhou,gzxu@sjtu.edu.cn}

Abstract

Mobile Ad-hoc Network (MANET) is a multi-hop wireless network without a preinstalled infrastructure, which is attractive for commercial use due to its nature of low cost and fast deployment. Routing and security are two major challenges in building a real commercial MANET environment. In this paper, we focus on the software solutions to implement a secure and efficient multi-hop MANET platform. We discuss an efficient and stable model based on on-demand routing protocols that is designed and implemented on Windows CE with special considerations for security.

1. Introduction

Mobile Ad Hoc Network (MANET) [1] [2] is a multi-hop wireless network without a preinstalled infrastructure. Mobile nodes that are within each other's radio range communicate directly via wireless links, while those that are far apart rely on other nodes to relay messages as routers. Such network first arose in disaster rescue and battlefield operations due to its flexibility and relatively low cost. Nowadays, with the prosperity of consumer electronic devices, such as PDAs, mobile phones, set-top boxes, embedded devices in cars, there is a trend to adopt ad hoc networks for commercial uses.

Windows CE [8] is a commercial embedded operating system designed for devices with small amounts of memory, storage and CPU processing power. It uses a component-based structure so that Original Equipment Manufacturers (OEMs) can choose only the operating system features that they require for their specific hardware platform. Windows CE also supports many multimedia applications and games. In recent years many efforts have been put on implementing MANET on Windows CE.

Routing model and security problem are two major problems when applying ad hoc networks. The infrastructure-less and dynamic nature of MANET demands a new set of routing protocols. Existing protocols to solve this problem include, but are not limited to AODV [9], DSR [12], DSDV [13] and TORA [14]. Most of the existing studies of these protocols are only discussed in theory. Most implementations to date are for Linux, and mainly for experimental purposes. Reports on implementations in Windows CE.NET for commercial use are much less. On the other hand, security [3] in MANET is a hot topic because the salient features of MANET pose new challenges in securing the communication. Canned security solutions like IPSec [15] [21] are not applicable.

In this paper we present the solutions to above-mentioned issues from the point of view of the system-level software implementation. Focus has been put on the software solutions to facilitate a secure and efficient multi-hop MANET platform. We hope our work will contribute to the proliferation of MANET in real commercial world.

2. Existing implementations and related work

There have been several implementations of ad-hoc routing protocols. In this section, we provide a comparison on these implementations.

V. Kawadia [4] studied the common requirements imposed by MENET and proposed a general framework with a generic API to facilitate MANET protocol implementations and deployments. It is known implementing the API normally require kernel modifications. As a variation, they first provided an implementation for Linux as a shared user-space library called ASL(the ad-hoc support library) using

the standard Linux 2.4 kernel facilities and then developed a full-fledged implementation of the AODV protocol, called AODV-UIUC[7] using ASL. Although this project was on Linux platform, the major principles are still suggestive and helpful to our design on Windows CE.

OLSR for Windows 2000 and Pocket PC is based on the PICA library [16] [17]. The PICA (protocol implementation specific library) serves to ease the work of transplant of the Ad hoc Protocol on Linux to the Windows 2000 as well as Windows CE platforms. One flaw of the PICA is that it may introduce overhead.

UoBWinAODV [6] is a project for Windows XP. It includes an NDIS intermediate driver and a user space program to manage the operations of AODV. However, UoBWinAODV meet with the problem of low efficiency. For example, each data packet to be sent out must search the matched record respectively in two similar tables, one in the IP layer and the other in the NDIS intermediate layer. In our model, we avoid such repeated search in a long list by some improvements in the NDIS driver.

WinAODV by Intel is similar to UoBWinAODV generally and differs in the way that it can be used together with 'PUDL', which assists WinAODV to choose the best quality neighbor links.

Self-Organizing Neighborhood Wireless Mesh Networks [5] for Windows XP is a community-based multi-hop wireless networks implemented by Microsoft networking research group, with features including self-stabilizing, multi-path multi-hop routing, auto-configuration and link quality measurement, etc.

The implementations mentioned above are all meant to address the on-demand routing problems. Some of them, such as ASL and PICA, provide a generic interface. Others are implemented for specific operating systems. Here we only list the projects for Windows, while those for Linux were enumerated and analyzed by V. Kawadia in [4].

We also note that there were some projects reported that support QoS like link quality measurement and adaptive link choice, but few are aimed at embedded operating systems like Windows CE with consideration of security issue in practice as we present in this paper.

3. Routing model

3.1. Routing protocols

In MANET, each node is capable of acting as a router to provide end to end communication through multi-hops. Most ad-hoc routing protocols can be classified into two categories: proactive routing and reactive routing. Proactive (or table-driven) routing protocols are similar to those in the wired world, which maintain routes to all possible destinations by periodically exchanging control messages. Reactive (or on-demand) protocols are designed specifically for ad-hoc networks, which discover routes only when there is such a demand for it. Obviously the second mechanism requires less control messages.

Ad Hoc On-Demand Vector Routing (AODV) protocol [9] is a typical on-demand routing protocol for ad hoc mobile networks. The routing messages do not contain information about the whole route path, but only about the source and destination. Therefore, the size of routed messages does not grow on the routing path. It utilizes destination sequence numbers to specify how fresh a route is and avoid loop freedom. Experiments have shown that AODV performs satisfactorily in most cases.

In AODV, a node updates its Neighbor List by HELLO message, which is periodically broadcasted from every node with lifespan equals to 1. Whenever a node needs to send a packet to a destination for which it has no fresh enough route, it broadcasts a route request (RREQ) message to its neighbors. Each node that receives the broadcast sets up a reverse route towards the originator of the RREQ. When the intended destination receives the RREQ, it replies by sending a Route Reply (RREP). The RREP is unicasted back to the originator of the RREQ. At each intermediate node, a route to the destination is set. Optionally, a Route Reply Acknowledgment (RREP-ACK) message may be sent by the originator of the RREQ to acknowledge the receipt of the RREP. In addition to these routing messages, Route Error (RERR) message are used to notify the other nodes that certain nodes are not anymore reachable due to a link breakage.

3.2. Available solutions for Windows CE.NET

We conclude from the above projects, which most on-demand routing protocols (like AODV) require the following services:

a) Identify the need for a route request.
b) Notify ad-hoc routing daemon of a route request.
c) Queue outstanding packets waiting for route discovery.
d) Re-inject outstanding packets after successful route discovery.
e) Update the routing cache.

However, most of current operating systems including Windows CE.NET do not provide such

services explicitly. To address these problems, we first propose four major methods for comparison.

The first method is to port the existing AODV implementation for Linux to Windows CE.NET using PICA mentioned in Section 2. It is an easy and straightforward, but overhead and time delay will be introduced

The second method is to change the kernel of Windows CE.NET so that it can support the MANET protocols directly. It is a long-term fundamental solution. However, since the codes for IP layer in Windows CE.NET are not open to the public, we do not apply this idea in this stage.

The third method is to adopt an NDIS intermediate (IM) driver and a user routing program. The Network Device Interface Specification (NDIS) is a public interface, which governs the communication between interface device drivers controlling hardware adapters. The NDIS intermediate driver is located between the NDIS miniport drivers and NDIS protocol drivers for some extended functions such as filter or encryption/decryption (Figure 1). The user routing program manages the routing environment as required by the AODV protocol. It is thus up to the user to decide when to turn on or turn off the routing program.

The fourth method is to implement all the functions in the NDIS intermediate driver. This method is feasible because Windows CE removes the barrier between kernel space and user space for device drivers. As all the networking device drivers effectively run in the protected user mode, they can be linked with the WinSock DLLs directly and send the AODV control packets as UDP datagram from port 654 to other host using WinSock functions. One drawback of this method is that it will increase complexity for users to control the routing model because it is in essential a driver rather than an executable file.

Table 1. Comparison of four methods

Method	Advantages	Disadvantages
Port AODV using PICA	Easy to implement	Overhead Time delay
Based on the modified kernel	A long solution A new generation of OS	Infeasible now Relative codes are not open.
An NDIS intermediate driver and a user routing program	Efficient and feasible Easily turn on/off by users	
All in the NDIS intermediate driver	Efficient and feasible	Difficult to control by users

Based on the above analysis, we set out our implementation based on the routing model using the third method.

3.3. Routing model

Figure 1 illustrates the structure of this implementation. Two main components are the AODV routing daemon and the NDIS IM driver, both of which run in the user space. AODV is implemented as an executable (.exe) file so that the users can choose to run it only when needed. NDIS IM driver is a dynamic link library (.dll) file, which is loaded and bound to the specific net card according to the registry when the operating system starts up. These two components share the same memory space for outstanding packets (data packets without any valid route), control packets and route usage information. They synchronize with each other using a pair of sophomore. AODV synchronizes its internal route cache with the IP route table by reading and updating the IP route table using the IP Helper tools. AODV sends out packets using WinSock DLLs.

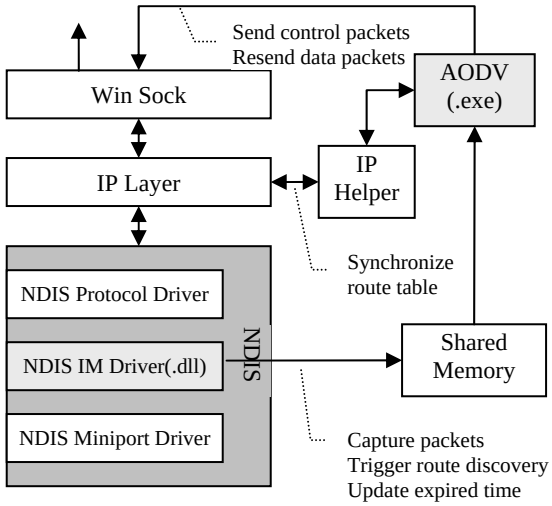


Figure 1. Software architecture of routing

We find that the Win Sock DLLs in Windows CE.NET do not support 'raw' socket. This adds to the complexity for implementing AODV when it re-sends the outstanding packets, because the transport layer type and port of those packets vary over different applications. Figure 2 illustrates our solution to this problem.

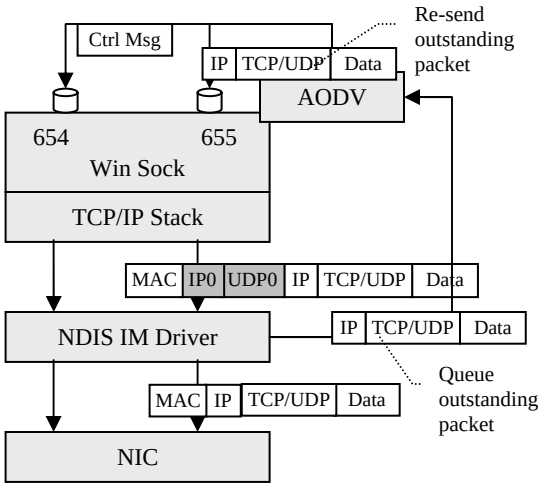


Figure 2. Outstanding Packet Flow

UDP port 654 is the reserved port for AODV. To be compliant, we define UDP port 655 for the re-injected data packets. Therefore, NDIS IM driver could distinguish the re-injected data packets from the control packets by their UDP port number. As shown in Figure 2, queued outstanding packets in AODV include the source IP header, the TCP/UDP header and data. After successful route discovery, we re-inject the outstanding packets and send them through UDP port 655 without consideration of the source TCP/UDP type or port number. Hence, all parts of the source outstanding packets are regarded as payload by the TCP/IP stack. New UDP header (UDP0) and IP header (IP0) are added as well as the MAC header. While in the NDIS IM Driver, the redundant headers IP0 and UDP0 are cut and re-packed. Although the re-pack operation in the NDIS IM Driver is a little time-consuming, it will not impact the whole performance because this only affects re-injected outstanding packets.

3.4. Improve the search efficiency

All the improvements described below can be applicable in a simple, flat and trusted MANET environment without sub-net or compromised nodes.

We find out that the bottleneck of the model lies in the search algorithm, which is used both in the send filter function of NDIS IM driver to filter outstanding packets as each packet passes down and in the AODV routing daemon when processing the route control packets and outstanding packets. We reduce the search cost in two ways.

Since the search in the NDIS IM driver occurs for each packet, a better solution is to replace the process of searching and comparing with a single comparison. We improve the searching effectiveness by setting a static ARP record in the system and comparing the MAC address rather than IP address in the NDIS IM driver.

In the original version, route daemon updates not only the IP route table but also the send filter list in NDIS intermediate driver. When a data packet passes the NDIS intermediate layer, its destination IP address is compared with those in the send filter list. If there is no match, then this data packet is regarded as an outstanding packet and should be sent to routing daemon.

In the improved version, we assume that all the mobile nodes in the MANET have the same *virtual gateway*, which does not represent a concrete device. Hence, the next-hop of default route entry in the IP route table is the IP address of the virtual gateway.

Dest:	Nets-hop:	Interface:
0.0.0.0	10.2.3.1	10.2.3.6

We set a new ARP entry. The default gateway address maps a pre-defined MAC address VMAC (ie.00-00-00-00-00-00).

Ip Addr:	MAC Addr:
10.2.3.1	00-00-00-00-00-00

Thus, when a data packet passes the IP layer without a route match, it is then by default set to the IP address 0.0.0.0 entry, specifying that the destination MAC address of any outstanding data packet is VMAC. So, in the NDIS IM driver, the data packet only need to compare its destination MAC address with VMAC rather than the IP address filter list that is much longer.

The second improvement is to choose a more efficient search algorithm in the AODV routing daemon. In a flat MANET with no sub-net, the conventional linear search of hash tables is suitable here because there is no need for prefix matching.

4. Security solutions

4.1. Security issues in MANET

As Zhou indicates in [3], there is an increasing possibility of eaves-dropping, spoofing, and denial-of-service attacks on MANETs, due in part to their poor physical protection and dynamic topology and dynamic membership. Some security-enhanced versions of ad-hoc routing protocols, such as SAODV[10], SRP[18], ARAN[19] and SEAD[20], have been proposed to ensure the operation of the routing protocol unaffected by attempts to forge or alter the routing protocol control messages.

Meanwhile, data packets could be protected by conventional encryption methods.

However, Public Key Infrastructure (PKI), or more basic key exchange techniques are difficult to implement in an ad hoc network due to the lack of authorities of trust and appropriate network infrastructure [11]. Many schemes have been proposed, yet a full-fledged and feasible key management solution is still underway. In our work, we suggest and demonstrate that the key distribution approaches could be tailored for Ad Hoc network to different scenarios.

4.2. Security services provided by Windows CE.NET

Windows CE supports Secure Sockets Layer (SSL) security protocols for secure network communication. SSL supports are available directly from WinSock. These applications use secure sockets to send and receive encoded data over the communication lines.

IPSec is designed to encrypt data as it travels between two computers, protecting the data from modification and interpretation. It is a key line of defense against internal, private network, and external attacks. Windows CE-based IPSec does not provide programming interfaces for developers.

Cryptography is employed to ensure data integrity and secures communication in the Windows CE-based applications. The Microsoft cryptographic system consists of application, operating system (OS) and cryptographic service provider (CSP). Applications communicate with the OS through the cryptographic API (CryptoAPI). The OS communicates with CSPs through the cryptographic service provider interface (CryptoSPI).

WindowsCE.NET also provides security services for user authentication, credential management, and message protection through a programming interface called the Security Support Provider Interface (SSPI). Within SSPI, different security providers are available, such as the NTLM security support provider (SSP) and Kerberos SSP. Each one contains different authentication and cryptographic schemes.

4.3. A secure routing model

In this section, we integrate our security solutions into the routing model described in Section 3. It is assumed that ad hoc nodes have permanent addresses and are in a flat MANET without sub-nets (E.g., the net meeting scenario). Therefore, bindings between public keys could be achieved by free distribution or by a certification authority (CA) to issue public key certificates. Figure 3 illustrates the structure of our secure routing model.

SAODV (secure AODV) is a secure version of basic AODV routing protocol, which achieves those security goals: import authorization, source authentication and integrity. Two mechanisms are therefore used: digital signatures to authenticate the non-mutable fields of the message, and hash chains to secure the hop count information (mutable field of the message). In Windows CE.NET, Crypto APIs are used to provide the required digital signature and hash chain functions.

IPSec is used to secure the network data transmissions in MANET. It is necessary that the IPSec policy will be able to apply certain security mechanisms to the data packets and just bypass the routing packets, which typically can be identified because they use a reserved UDP port number. However, Windows CE-based IPSec does not provide programming interfaces for developers to perform such specific operations.

Hence, the Windows CE-based IPSec is disabled and some IPSec functions are embedded in the NDIS IM driver. Detailed operations are described with the pseudo-code (see Figure 4 and Figure 5).

Furthermore, some sensitive applications may choose the secure socket (support SSL) to transmit data.

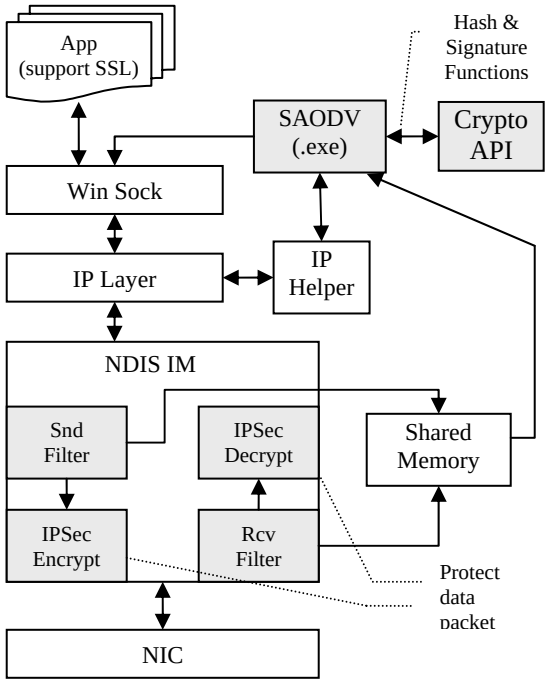


Figure 3. Structure of a secure routing model

```
if  (UDP & UPD port == 654)
then {     block,
           send to AODV} //route control message
else  {    decrypt,
           pass up} //data packet
```

Figure 4. Pseudo-code of NDIS IM receive part

```
if (MAC == Default_MAC)
then  { block,
         send to AODV} //packet with no route
else  if  (UDP & UDP port == 654)
      then  { pass down } //route control message
      else  if  (UDP & UDP port = 655)
            then  { cut IP0 and UDP0 from the packet,
                    encrypt,  //if use IPSec
                    record the route used, //inform AODV
                                           //periodically
                    pass down} //resend data packet with
                               //valid route
             else  { encrypt,  //if use IPSec
                    record the route used, //inform AODV
                                           //periodically
                    pass down} //other packets with valid
                               //route
```

Figure 5. Pseudo-code of NDIS IM send part

5. Conclusion and future work

In this note, we study several alternative implementations of on-demand routing protocols in Windows CE.NET platform, and propose an efficient routing model in a specific embedded operation system. Then, we explore the security problems for MANET, and integrate our security solutions into the routing model with the help of the security services provided by Windows CE.NET.

We believe that routing and security are two major issues in building a real commercial MANET environment. In this paper we present the solutions to those issues from the point of view of the system-level software implementation. With our secure routing model embedded, several mobile nodes could form a secure and efficient multi-hop MANET, in which some application scenarios like net-meeting could be realized. We hope our work will contribute to the proliferation of MANET in real commercial world.

In the future, we plan to extend our work to more complicated yet more practical schemes, such as dynamic address allocation, auto configuration and etc. The corresponding key management solutions, which are laid aside for the moment, will be put into further study.

6. References

[1] K. Kuladinithi and C. Gg, "Tutotrial on mobile ad hoc networks", in *Proc. of 1st Regional Conference on ICT and E-paradigms*, June 2004.

[2] Manet working group charter. [Online]. Available: http://www.ietf.org/html.charters/manet-charter.html

[3] L Zhou, Z J Haas. "Securing Ad Hoc networks", in *IEEE Network Magazine*, 1999,13(6):24~30

[4] V. Kawadia, Y. Zhang, and B. Gupta., "System services for ad-hoc routing: Architecture, implementation and experiences", in *Proc. of the 1st Int'l Conf. on Mobile Systems, Applications, and Services (MOBISYS)*, May 2003.

[5] Self-organizing neighborhood wireless mesh networks. [Online]. Available: http://research.microsoft.com/mesh/

[6] Uobwinaodv implementation. [Online]. Available: http://www.aodv.org/

[7] Aodv-uiuc implementation. [Online]. Available: http://sourceforge.net/projects/aslib/

[8] MSDN. [Online]. Available: http://msdn.microsoft.com/

[9]C. E. Perkins, E. M. Royer, and S. R. Das. "Ad hoc on-demand distance vector (AODV) routing". *IETF INTERNET DRAFT, MANET working group*, Jan. 2002. draft-ietf-manet-aodv-10.txt.

[10]M. Guerrero. "Secure ad hoc on-demand distance vector (SAODV) routing". IETF MANET Mailing List, Message-ID 3BC17B40.BBF52E09@nokia.com, http://www.cs.ucsb.edu/eroyer/txt/saodv.txt , Oct. 2001.

[11]N. Asokan and P. Ginzboorg. "Key agreement in ad-hoc networks". *Computer Communication Review*, 23(17):1627–1637, Nov. 2000.

[12]D. A. Maltz. *On-Demand Routing in Multi-hop Wireless Mobile Ad Hoc Networks.* PhD Thesis, Carnegie Mellon University, 2001.

[13]C. E. Perkins and P. Bhagwat. "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers". In *Proceedings of ACM SIGCOMM'94*, London, U.K., Sept. 1994.

[14]V. D. Park and M. S. Corson. "A highly adaptive distributed routing algorithm for mobile wireless networks". In *Proceedings of IEEE INFOCOM*, 1997.

[15] S. Kent and R. Atkinson, "IP Authentication Header," *IETF RFC 2402*,1998.

[16] C. M. T. Calafate and P. Manzoni, "A multi-platform programming interface for protocol development". *Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing*, 2003, Genova, Italy

[17] C. M. T. Calafate, R. G. Garcia and P. Manzoni, "Optimizing the implementation of a MANET routing protocol in a heterogeneous environment". *Eighth IEEE International Symposium on Computers and Communications*, 2003, Kemer-Antalya, Turkey

[18]P. Papadimitratos and Z. J. Haas, "Secure routing for mobile ad hoc networks". *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, Jan 2002.

[19]B. Dahill, B. N. Levine, E. Royer, and C. Shields, “A secure routing protocol for ad hoc networks”. Technical Report UM-CS-2001-037, University of Massachusetts, Departament of Computer Science, Aug. 2001.

[20]Y. C. Hu, D. Johnson, and A. Perrig, “SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks”. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA ’02), June 2002*, pages 3-13, June 2002.

[21] S. Kent and R. Atkinson, “IP Encapsulating Security Payload (ESP),” *IETF RFC 2406*, 1998.

A linux based Bluetooth scatternet formation kit: from design to performance results

Francesca Cuomo, Andrea Pugini
INFOCOM Dept., University of Roma "La Sapienza", Via Eudossiana 18, 00184, Rome (Italy)
cuomo@infocom.uniroma1.it, anpugini@libero.it

Abstract

Bluetooth is a key technology for the deployment of wireless personal (WPAN) and body area (WBAN) networks. This paper describes an implementation experience of a scatternet formation protocol on a Bluetooth stack for Linux (BlueZ).

The innovative elements of this work are: I) the development of two software packages designed to create and manage multi-hop Bluetooth wireless networks, based on simple USB Bluetooth devices; II) the analysis of results on this real implementation.

The formed network (scatternet) presents limited formation delay and interesting performance in view of personal/body area applications. Implementation details as well performance results are discussed.

These results have been derived in the framework of the FIRB VICOM project founded by the Italian Telecommunication Ministry.

1. Introduction

Bluetooth is a short range, low power and low cost wireless technology for ad-hoc networking. Eight Bluetooth devices are interconnected in a piconet and share the same radio channel which supports about 1 Mbit/s [1] nominal symbol rate. Class 3 devices have a transmission range of about 10 meters. No network infrastructure is envisaged: self-organization and peer communications lead to a complete ad-hoc connectivity. Piconets are dynamically set-up and torn down. In a piconet two Bluetooth devices (named also nodes) exchange information by means of a master-slave relationship. Master and slave roles are dynamic: the device that starts the communication is the master, the other one is the slave. A master can connect with up to 7 slaves. Multiple access in a piconet is centrally regulated by the master that adopts a polling scheme on a slotted time structure. Piconets can coexist in the same area and interconnect in a scatternet.

A Bluetooth device that joins more than one piconet is named bridge and participates to communications in different piconets on a time-division basis. A device can play the role of master in only one piconet. Scatternet formation in Bluetooth has recently received a significant consideration [2]. Existing solutions can be classified in single-hop ([3][4][5]) and multi-hop ([6][7][8][9]).

In this paper, we report our experience using the Bluetooth stack for Linux (BlueZ). We developed innovative software applications with graphical user interface, called J-BlueZ and BASkit. The former translates visual inputs in BlueZ commands, the latter supplies automata to dynamically form and manage scatternets. Our study includes testing and performance evaluation of these packages to collect relevant results about implemented scatternet formation protocol. These packages are used for mobile immersive communications to allow a user, managing simple devices (e.g., PDAs and smart tags), to enter and act in pervasive environments.

2. The Bluetooth Profiles

The Bluetooth standard defines all the protocols from Bluetooth Radio layer to upper network layers. Also different profiles are specified [10][11].

The Personal Area Networking (PAN) profile describes how two or more Bluetooth enabled devices can form an ad-hoc network and how the same mechanism can be used to access a remote network through a network access point. The profile specifies the roles: the Network Access Point (NAP), the Group Ad-hoc Network (GN), and the Personal Area Network User (PANU). NAP can be a traditional LAN data access point while GN and PANU are master and slave nodes in a piconet.

The PAN profile defines a means of enabling Bluetooth devices to participate in a personal area network. Completely un-modified Ethernet payloads can be transmitted using the Bluetooth Network

Encapsulation Protocol (BNEP) to exchange packets between Bluetooth devices (Figure 1).

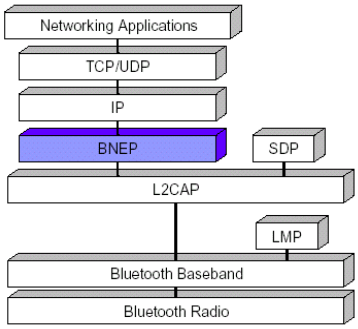


Figure 1: BNEP profile

This profile defines how PAN is supported in the following situations:

- Ad-hoc IP networking by two or more Bluetooth devices in a single piconet;
- Network access for one or more Bluetooth devices.

BNEP specifications describe the protocol to be used by the Bluetooth PAN profiles. They also define a packet format for Bluetooth network encapsulation used to transport common networking protocols over the Bluetooth media. Bluetooth network encapsulation supports the same networking protocols that are supported by IEEE 802.3/Ethernet encapsulation. Packets from the supported networking protocols are contained in Bluetooth network encapsulation packets, which are transported directly over the Bluetooth L2CAP protocol.

3. BlueZ Description

Many implementations of the Bluetooth stack are available, but not all of them are free or open source.

Open source implementations are free with easy access to the API's and source code. General Public License makes sure that the software is always designed in a open manner and this may guarantee many open source developments in the future.

BlueZ is the Official Linux Bluetooth protocol stack for the Linux Kernel [12].

Some of BlueZ's features are (Figure 2):

- Support for core Bluetooth layers and protocols;
- Support for multiple Bluetooth devices;
- Standard socket interface to all layers;
- Multiplatform: x86 (single and multiprocessor), SPARC, ARM, DragonBall;
- HCI UART, USB, PCMCIA and Virtual HCI drivers;
- Support for L2CAP, SCO, RFCOMM and SDP;
- Support of GAP, DUN, LAN, SPP, PAN profiles;
- OBEX Object and File Push Profile.

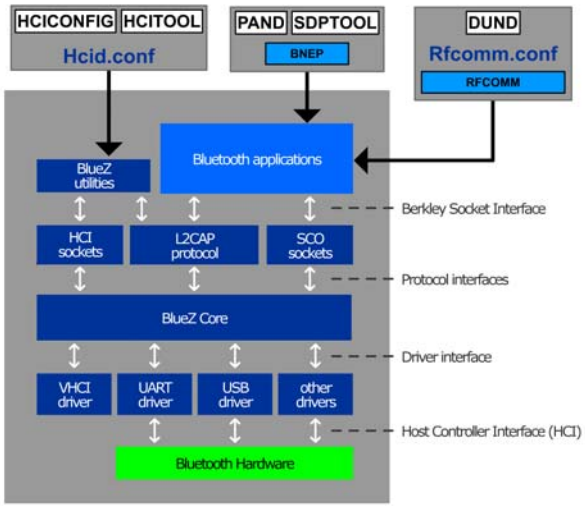


Figure 2: BlueZ stack

BlueZ is distributed in a set of packages . BlueZ core package, called "bluez-kernel", includes all functionalities to set up the core of Bluetooth. It takes care of making HCI-devices, the L2CAP and LMP/LC protocols. It is already a part of 2.4.21 kernel or higher. [12]. Other packages complete BlueZ: "bluez-libs", "bluez-utils", "bluez-sdp", "bluez-pan", "bluez-hcidump", "bluez-hciemu", "bluez-bluefw".

3.1. BlueZ tools

BlueZ contains three important tools to manage Bluetooth devices.

The first BlueZ tool is *hciconfig* that makes possible to configure device at the HCI level.

The *hciconfig* main call for the first Bluetooth interface is 'hciconfig hci0'; this acts like the 'ifconfig' network command in Linux. To set up a device, the first call must be: '# hciconfig hci0 up'.

Another tool is *hcitool*. This tool is employed to use the main features of device at HCI level (e.g., inquiry and connecting calls).

Third tool is PAND, treated in the following.

3.2. PAND

BlueZ package implementing PAND contains the necessary to use two of the most used profiles in Bluetooth: LAN and PAN. The PAN profile specifies the three Bluetooth roles:

- Network Access Point (NAP): acts as proxy, router or bridge between an existing network infrastructure (typically LAN) and (up to 7 active) wireless clients (PANUs);
- Group ad-hoc Network (GN) controller: entity interconnecting up to 7 (active) PANUs (master in the Bluetooth Piconet);
- PAN User (PANU): client of a NAP or client-type member of a GN. It corresponds to the slave role of the Bluetooth Specifications.

Both NAP and GN are service providers that coordinate the traffic in the PAN. NAP is also an access point to another network. The PANUs, on the other hand, are clients or users of the PAN Service.

The Bluetooth PAN feature offers (among other features) IP support over Bluetooth (i.e., L2CAP).

3.2.1. PAND: General Setup. We consider the configuration of a single link. The master of the link is configured as the GN of the link, the slave as the PANU. The basic idea behind PAND is that you start it on one side in a server fashion (configuring the master side of the link), using the '--listen' parameter and then you can establish an explicit connection between PANU and GN.

At the end of the PAND configuration, each interface activated between two nodes represents a wireless link and is named 'bnepx' (with x=1,2,3..).

3.2.2. PAND interfaces. After the PAND connection is established, a virtual network interface 'bnep0' is created on both nodes. This interface can be configured using 'ifconfig' (e.g., '# ifconfig bnep0 10.0.0.1') to associate the IP address of the node to the interface. The handling of the 'bnepx' interfaces could become complicate in case of multihop paths. In this case, a node belonging to a path should receive from an interface (e.g., 'bnep2') and forward on a different interface (e.g., 'bnep4'). This means that the node needs a routing table and this table must be dynamically changed when new connections are established in the scatternet.

3.2.3. PAND Ethernet Bridging. In order to simplify the aforementioned situation, it is possible to use the "802.1d Ethernet Bridging" contained in the 2.4.x Linux kernel. The idea of this feature is to associate the multiplicity of Bluetooth layer 2 network interfaces ('bnepx') to a single layer 3 interface named 'pan0'.

Each device associates its IP address to this interface, and associates all the communications related with the interface 'pan0' to the BNEP protocol. This procedure has to be repeated at each node for each interface 'bnepx' created by PAND.

Once the 'pan0' interface is created it is the unique interface to link any 'bnepx' interface. The composition of the different 'pan0s' in the nodes constituting the piconet appears as a unique "piconet broadcast" interface. Then, to use the scatternet as a unique broadcast domain you need to associate the BNEP broadcast protocol to the pan0.

4. Modules and Interfaces using BlueZ: from J-BlueZ to BASkit

BlueZ supplies a large range of services and profiles to create and manage Bluetooth ad-hoc connections but requires to act single commands using shell terminal. Such procedure is slow and complex: it requires complete knowledge of BlueZ sintax and semantic. To facilitate the use of BlueZ in a scatternet test-bed framework, we developed a Java Graphical User Interface (GUI) for BlueZ: J-BlueZ. This is an intuitive, essential, graphic interface with basic capabilities over BlueZ. It basically does not implement any network formation protocol but gives control to the user, merely transforming visual commands in series of BlueZ commands. J-BlueZ executes commands invoking shell terminal and providing to it a string containing commands. As it happens when the interface is executed by human user.

Beside the GUI, a second software package was developed: BASkit (Bluetooth Ad-hoc Scatternet kit). This package is a network module providing J-BlueZ simplicity and effectiveness but implementing a concrete Bluetooth ad-hoc network formation protocol. The goal is twofold: create dynamic and optimized ad-hoc Bluetooth network, and manage and monitor connections in this network. It supplies multi-hop connectivity and supports TCP and UDP sockets, IP broadcast and ICMP to upper layers.

Therefore, J-BlueZ and BASkit can be used to get connectivity with a Bluetooth device and present other relevant advantages for network managers and deployers: J-BlueZ can be easily used being unaware of components as operative system or BlueZ; BASkit can be also used in a test-bed to verify implemented Bluetooth ad-hoc network formation protocol and relevant performance.

4.1. J-BlueZ

4.1.1. Usage. The J-BlueZ application module was designed to be a simple and intuitive tool to create and

manage network connections. According to this guiding principles, the interface provides the core actions:

- collecting available devices information;
- connecting to/ disconnecting from a device;
- setting up parameters.

User can act using buttons and text fields creating and tearing down connections by himself or select one, or more, of "Automatic" features and let application create and manage network connections as a wizard.

4.1.2. Structure. J-BlueZ structure is designed to accomplish few essential goals:

- to display available devices information;
- to make possible connection and disconnection to/from a remote device.

While working, the application uses some parameters to lead its behavior, user can modify these parameters and set some procedures as autonomous or not.

J-BlueZ has a unique main menu (Figure 3) containing:

- device information area, showing available devices data;
- a series of buttons and text fields to input commands.

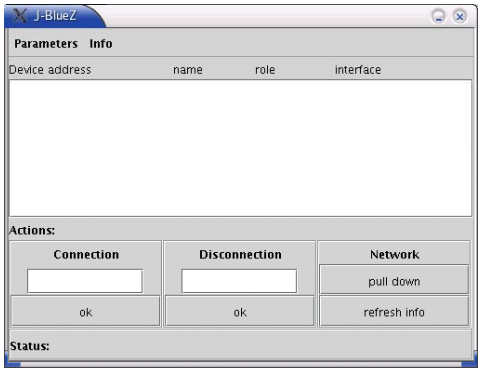


Figure 3: J-BlueZ main window

In device information area, a row for each available device shows: the address, the name, the connection role and the BNEP interface (if device is connected).

Parameters menu is in the upper left corner of the window (Figure 3): each menu item activates a dialog window to change parameter value. "Time interval" item can be used to input a value for the time interval of the periodic environment inspection: typing '0' (zero) periodic environment inspection is disabled. In this dialog box the user can also activate environment inspection after every command execution. Another feature is "Automatic Behavior": this radio button enabled, the module acts as an autonomous tool, as describe in the following.

4.1.3. Behavior. J-BlueZ is a general purpose module, its main goal is to simplify the execution of PAND commands and the inspection of network information providing the user with basic options to do so. Application does not implement any network formation protocol by itself.

When application is launched it enables potential incoming connection from other devices. This phase is repeated periodically and continues during all the life time of J-BlueZ. In every moment the user can decide to connect to a device and become, in this way, a slave of that device. Therefore, many connections are possible at the same time with different roles for a device. Commercial devices employed in our test-bed support multiple L2CAP and multiple roles.

J-BlueZ does not add any additional control features on connection: if too many connections are attempted J-BlueZ will simply communicate an error. In case of failure, short messages are displayed and information about environment and network status are not updated.

Information about available devices are provided after environment inspection procedure and are maintained in the devices text area until new environment inspection is performed. Environment inspection is performed:

- each time user clicks the "refresh info" button,
- each time periodic inspection time interval expires (if it is greater than zero),
- each time a user command is executed and "Automatic Inspection" is selected.

During inspection, as all other procedures, application appears as "freezed": buttons and text fields are inactive and information do not change anymore. Only when the procedure is accomplished information are updated and buttons are reactivated.

In order to create connections an IP address must be assigned to the Bluetooth device by the J-BlueZ application. At the launch, in the startup dialog, the application asks for this address and the device name.

As we mentioned before an automatic behavior is possible. Activating this feature makes J-BlueZ a completely autonomous connection wizard: "Connect" and "Disconnect" buttons are disabled and application automatically inspects environment, using defined time interval, and creates connections to every available device. In this way the host is eventually linked to each device creating more links than necessary i.e., generating a redundant topology.

J-BlueZ simplicity and effectiveness is therefore preferred in the tradeoff with network efficiency. This network efficiency is delegated to the BASkit module as described in the following.

4.2. BASkit

BASkit (Bluetooth Ad-hoc Scatternet kit) includes two logical entities:

- network module for dynamic and optimized ad-hoc Bluetooth network formation and management;
- Graphical User Interface for visual direct control and action capability over entire network.

The GUI is not strictly linked to network module: the latter must be first launched, or loaded in background as service, and then GUI can be activated by a shell command or by double clicking on its icon.

4.2.1. Network module. Network module creates and manages the network, providing "plug-and-play" connectivity.

The network module acts in two steps:

- it adds new nodes to the network using a fast and efficient approach;
- it manages and organizes the network topology by applying a suitable algorithm.

This algorithm, called SHAPER [8], ensures tree-like network topology and it enables the user to add new nodes to the network, merge different trees, reorganize automatically the network topology after nodes abandon event (so called self-healingness). To do so it uses short messages to and from nodes, sent over the network through specific sockets.

When a host activates BASkit (network module) it attempts to connect to other devices, that is already existing network, by:

- applying the J-BlueZ "inspection" procedure,
- connecting to one device among those discovered, randomly chosen.

Such procedure can significantly reduce the time needed to be connected to the network, avoiding the long time required by multiple single link creation in BlueZ. Such delay is caused by Bluetooth usage of a single beacon channel and BlueZ specific Bluetooth implementation and therefore cannot be changed without modifying BlueZ code. The BASkit module overcome this issue by reducing the number of connection attempts without modifying any BlueZ module.

When a BASkit host (A) creates its first connection it receives from the connected host (B) tree-like structure information, according to SHAPER algorithm. At this step different scenarios can occur:

- Host "B" belong to a tree-like structure - "A" becomes a member of the already formed network and follows SHAPER rules and procedures;
- Host "B" is isolated - Hosts "A" and "B" create a single leaf tree-like structure and become a new piconet.

Once nodes are connected in a network, connectivity given by BlueZ enables use of sockets to send and receive messages required by SHAPER. BASkit network module is a "scatternet" manager since manage logical tree-like structure: it creates scatternet using piconets.

4.2.2. Graphical User Interface. The BASkit GUI can be activated by the user when he/she needs to have control on module activity or be aware of the network status (additional features to network connectivity). This interface provides complete network description and intervention capability, it is a key tool for network deployer and manager to control and to act on entire Bluetooth ad-hoc network from a single node.

The BASkit GUI development process followed some fundamental guidelines:

- we need detailed information about network and connections;
- we want to monitor module working, with its sockets, channels and messages;
- we want to be able to directly act on the network.

These requirements defined the GUI layout: graphic pane displaying visual network structure, text areas showing information about network nodes and module work, buttons and text fields to input commands. Some elements are inherited from J-BlueZ and other realized for the first time.

BASkit implements scatternet formation protocol in a tree-like form and uses this structure in the GUI to display scatternet nodes and node relative information (figure 4).The resulting Panel, even if giving in one shot many information, has a high intelligibility in any information displayed.

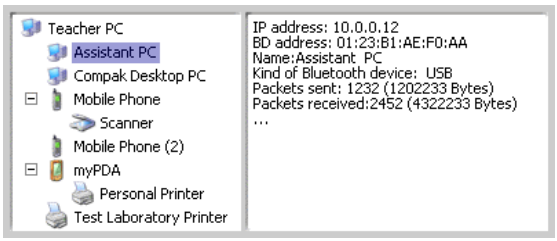


Figure 4: BASkit panel showing scatternet tree-like structure

Thanks to this simple and clear picture we identify piconets within scatternet. The highest node ("Teacher PC" in Figure 4) is the master of first piconet, it is the scatternet root. Devices listed below are slaves (and leaves). Devices master of other piconets are marked with "-" and have other devices listed below.

BASkit GUI, thanks to the text panel, provides information about the local Bluetooth host device features such as: type, address, status, traffic class, packet loss, other information distributed among the scatternet nodes.

Also BASkit interface has many J-BlueZ features, even if empowered, like connection, disconnection and network inspection, but procedures are implemented in different ways through different BlueZ instruments. Interface enables sending to remote node of information request and specific commands forcing it to execute operations. Manager, using such instrument, can send over network commands to remote nodes having complete control of network and nodes activity. There is a boundary to this control: remote nodes execute received commands only if their GUI is not active.

5. Experimental results

In this Section we evaluate:

- performance of the BlueZ stack with respect to link throughput, as well as latency, jitter and packet loss;
- performance of J-BlueZ and BASkit with respect to time required to create network.

5.1. BlueZ performance

We build up the experimental platform with actual Bluetooth commercial devices. The test hardware is composed by personal computers: two Pentium 4 CPU (1,8Ghz, Mandrake Linux 9.1) and one Pentium Pro CPU (800MHz, Linux 9.2). Commercial USB Bluetooth dongles, 3Com and Nortek brands, were used.

All the Bluetooth dongles can be bridges also with different roles, e.g. PANU/GN or PANU/PANU.

5.1.1. Piconet and Scatternet Performance Evaluation. We measured throughput on point-to-point connection between a master and a slave in a single piconet. Figure 5 shows the mean throughput for all possible Bluetooth packet encodings. The throughput evaluations have been carried out by executing an FTP of a file of 10 Mbytes.

Here, the number of slots used for transmission and the resistance to noise have a significant impact. Our result shows that the best throughput is achieved with DH5: 5 slots transmission and lower noise resistance (more data is transmitted than other codings).

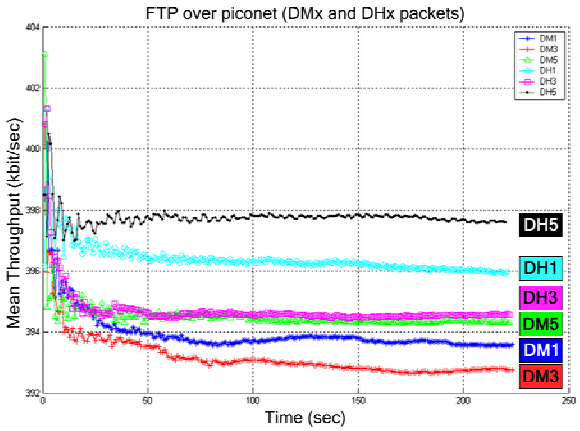


Figure 5: Mean throughput of a FTP over piconet

Note that the measured maximum throughput is far from the theoretical maximum (723 Kbit/sec). In fact, our BlueZ stack generates messages at different layers that take up bandwidth on the user interface.

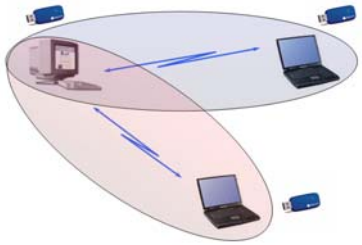


Figure 6: Topology of the scatternet used in the experiments

Then we measured throughput on point-to-point connection in different scenarios. In the first one we set up a piconet with one master and two slaves. In the second one, we have set up a scatternet connecting two piconets formed by two units (Figure 6). The bridge node plays different roles (GN – PANU) in the piconets. In the third case, we have two piconets, formed by master and slave, but bridge plays same role (PANU) in the piconets.

FTP mean throughput is: about 550 kbit/sec over 1-hop piconet, about 300 kbit/sec over 2-hops PANU-GN/PANU-GN scatternet, about 175 kbit/sec over 2-

hops GN-PANU/PANU-GN scatternet and about 60 kbit/sec over 2-hops piconet.

Scatternet tests have different throughput and this is caused by bridge node roles: a PANU-PANU bridge node has the same throughput in piconets but a GN-PANU bridge node asymmetrically allocates activity and causes a throughput reduction.

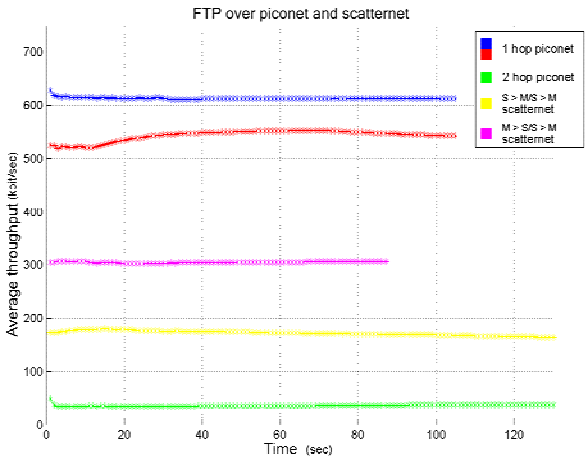


Figure 7: Mean throughput of a FTP over a two-hops piconet and scatternet

This experiment can primarily suggest to use, with BlueZ, multi hop scatternet topology rather than a single piconet. This result encouraged the design of suitable tool for the automatic scatternet formation (BASkit).

5.1.2. Audio-Visual Evaluation. A current challenge for video over a wireless link, using devices such as Bluetooth, is the high degree of variability in the radio signal strength meaning unreliable connections between devices. When sending video using Bluetooth connections, these unreliable connections can cause errors which result in huge defects in the quality of the image received. As for a two-hops connection, since the nominal mean throughput for a two-hops scatternet connection is 150 kbit/sec, it should represent a critic value. On the contrary, in case of two-hops connection over a piconet, the bit rate (60 kbit/sec) is not sufficient to support a videoconferencing application. From these considerations, we can notice that on a piconet built by a single link it should be possible to have a video-audio connection over Bluetooth without troubles.

We would like to notice that an intelligible video coding for video conferencing can be 100 kbit/sec, while a coding for an audio communication can be set up to 10 kbit/sec.

To evaluate the performance of the wireless network with Bluetooth, we have used the Gnomemeeting video-conferencing application over a two-hops scatternet. The video coding used had as maximum bandwidth 100 kbit/sec, while the audio coding is up to 10 kbit/sec. Gnomemeeting uses UDP packets. To have a term of comparison we also ran an FTP for a 10 Mbyte. We measured the packet loss and the jitter.

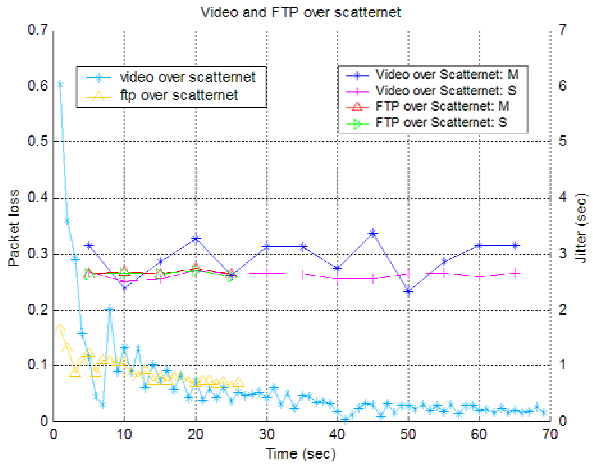


Figure 8: Packet loss of videoconferencing and FTP over two-hops scatternet

In Figure 8 we show the packet loss rate and the jitter both during the video conferencing and FTP. The Gnomemeeting application has a buffer size of 4 seconds to compensate the jitter. With reference of Figure 6, in the considered scenario the flows cross the GN/PANU bridge starting from a GN in a piconet and ending to the PANU in the other piconet (M case) or starting from a PANU in a piconet and ending to the GN in the other piconet (S case).

Results show that in case of videoconferencing, the packet loss is higher than in case of FTP application, while the jitter size remains quite limited (by the value of 3 seconds).

As for the perceived quality the audio results intelligible in all cases. The video presents clear images in the single link case, while in the two hops scatternet the quality decreases even if it remains acceptable.

5.2. J-BlueZ and BASkit performance

A key performance metric for J-BlueZ and BASkit is the time spent to establish a single link and to create

the whole network. To test J-BlueZ performance we use 3 nodes and measure the time elapsing from first module activation to last connection.

We have sequenced node activations with intervals of 20s, we have chosen 14s between start of environment inspection and connection trial, 30s between two automatic inspections. Performance is strongly biased by these values.

Nodes' name (A, B and C) reflects the activation order. Every node executes J-BlueZ, searching for available devices and then connecting to them. BlueZ 'scan' command could return a list with a subset of available devices. This problem, probably is caused by inquiry failure [7] and cannot be solved by J-BlueZ. Therefore J-BlueZ attempts to connect to a reduced number of nodes compared to available devices. J-BlueZ periodically repeats inspection and connection and eventually connects to newly discovered devices. At the end we have that all nodes are linked directly to all the other nodes (fully meshed topology).

Figure 9 presents three examples of formed networks. As for the time to establish single link connections we measured: a minimum value of 2s, a maximum value of 49s and a mean value of 38s. Measures on the time spent to create network indicate a minimum value of 78s, a maximum value of 89s and a mean value of 81s.

J-BlueZ and BASkit aim to different goals to form network: the former aims at linking a node to every available device, the latter at connecting a node to the scatternet.

BASkit has better performance than J-BlueZ since the topology to be formed is more rational: i.e., BASkit forms connected network without forming useless loops as happens in J-BlueZ (right part of Figure 9).

Thanks to the reduced number of links in the best case, when a single node enters an already formed network, only one connection is activated. On the other hand, in the worst case, a large number of isolated nodes entering concurrently the same area, perform a reduced number of connection trials since they stop as soon as the scatternet is connected.

10. Conclusions

This paper presents an experience of design and performance analysis of an experimental platform (built up with commercial devices) for Bluetooth ad-hoc networking. We implemented two suitable tools (namely J-BlueZ and BASkit) to connect, in a Linux environment, Bluetooth devices in a scatternet. Some key applications (data transfer and videoconferencing) have been tested on the formed network. Scatternet set-up delay has been also evaluated. Performance analysis showed the feasibility and simplicity in using our tools with satisfactory results.

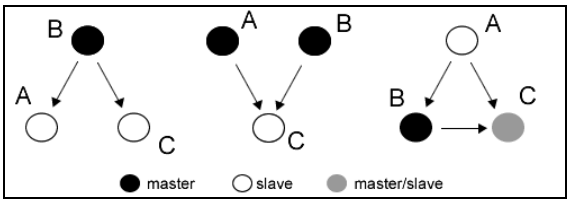


Figure 9: Three J-BlueZ test scenarios

10. References

[1] J. Haartsen, "The Bluetooth Radio System", *IEEE Personal Communications*, Vol. 7, n. 1, pp. 28-36, February 2000.

[2] P. Johansson, R. Kapoor, M. Gerla, M. Kazantzidis,"Bluetooth an Enabler of Personal Area Networking", *IEEE Network*, Special Issue on Personal Area Networks, pp. 28-37, September/October 2001.

[3] T. Salonidis, P. Bhagwat, L. Tassiulas, R. La Maire, "Distributed topology construction of Bluetooth personal area networks", Proc. of the *IEEE Infocom 2001*, pp. 1577-1586, April 2001.

[4] C. Law, A. Mehta, K-Y Siu, "Performance of a new Bluetooth scatternet formation protocol", Proc. of the *Mobihoc 2001*.

[5] G. Tan, A. Miu, J. Guttag, H. Balakrishnan, "An Efficient Scatternet Formation Algorithm for Dynamic Environments", in *IASTED Communications and Computer Networks (CCN)*, Cambridge, November 2002.

[6] G. Zaruba, S. Basagni, I. Chlamtac, "Bluetrees-Scatternet formation to enable Bluetooth-based personal area networks", Proc. of the *ICC 2001*, pp. 273-277, 2001.

[7] S. Basagni, C. Petrioli, "Multihop Scatternet Formation for Bluetooth Networks", Proc. of the *VTC 2002*, pp. 424-428, May 2002.

[8] F. Cuomo, G. Di Bacco, T. Melodia, "SHAPER: a Self Healing Algorithm Producing multihop bluetooth scattERnets", Proc. of the *IEEE Globecom 2003*, San Francisco, pp. 236-240 December 2003.

[9] F. Cuomo, T. Melodia, and Ian F. Akyildiz "Distributed Self-Healing and Variable Topology Optimization Algorithms for QoS Provisioning in Scatternets" IEEE JSAC, Vol. 22, N. 7, pp. 1120-1236 September 2004

[10] Specification of the Bluetooth System v 1.0 B, Volume 1, Core. Bluetooth Special Interest Group, December 1999;

[11] Specification of the Bluetooth System v 1.0 B, Volume 2, Profiles, Bluetooth Special Interest Group, December 1999;

[12] BlueZ Bluetooth Implementation for Linux, http://bluez.sourceforge.net/;

SESSION II

MANET Experimentations

An Experimental Study of P2P Group-Communication Applications in Real-World MANETs*

Franca Delmastro
CNR, IIT Institute
Via G. Moruzzi, 1 – 56124 Pisa, Italy
franca.delmastro@iit.cnr.it

Andrea Passarella
University of Cambridge, The Computer Laboratory
15 JJ Thomson Avenue – Cambridge CB3 0FD, UK
andrea.passarella@cl.cam.ac.uk

Abstract

Group-communication applications are a very promising opportunity for developing valuable MANET-based applications. However, real-world experimental studies are required to indicate the best solutions to implement them. We have implemented a real prototype in which alternative networking stacks can be used to support a distributed Whiteboard application. By means of experimental results we show that a standard P2P solution based on Pastry and Scribe is not suitable for MANET environments. We also show that a cross-layer P2P system optimised for MANETs (i.e., CrossROAD) is able to overcome many of the problems experienced with Pastry.

1 Introduction

Even though research on MANETs has been very active in the last decade, real applications addressed to people outside the research community still have to be developed. The typical simulation-based approach for the performance evaluation of MANETs is one of the main reasons of this. Often, simulation results turn out to be quite unreliable if compared to real-world measurements [1, 11], and real-world experiments are highly required for MANET applications to become reality, despite their high costs (in terms of time to set up) and intrinsic limitations (number of nodes).

By leveraging the self-organising nature of MANETs, group-communication applications can be an outstanding opportunity from this standpoint. In this paper, we focus on a significant example of this class of applications, and we evaluate complete networking solutions that could be used to develop it. Specifically, we consider the Whiteboard application (WB), which implements a distributed whiteboard among MANET users. WB allows users to share drawings, messages and other dynamically generated content. Such a self-organising distributed application can be naturally supported by P2P systems. In our prototype, WB uses Scribe [4] to share WB content among users via application-level multicast trees. Scribe requires a P2P overlay network based on a DHT. Our prototype includes two alternative P2P solutions, i.e., Pastry [13] and CrossROAD [8]. Both of them provide the same functionalities toward above layers through the P2P commonAPI [6], but CrossROAD is explicitly designed for MANET environments. It reduces (with respect to Pastry) the network overhead related to the overlay management by exploiting cross-layer interactions with a proactive routing protocol. Specifically, the CrossROAD implementation is compliant to the cross-layer framework described in [7]. Finally, the prototype includes OLSR [12] and AODV [2] at the routing level. Pastry performance is evaluated on top of both routing protocols while CrossROAD is evaluated on top of OLSR, since CrossROAD is designed to exploit a cross-layer interaction with a proactive routing protocol.

The main contribution of this paper is evaluating through real experiments *complete* networking solutions for developing distributed applications such as WB in real-world MANETs. We evaluate our prototype at two different levels, i.e., we quantify i) the QoS perceived by WB users, and ii) the quality of the multicast tree generated by Scribe. First of all, we show how a proactive routing protocol performs better than a reactive one with regard to this kind of applications. Then, we highlight that a solution based on Pastry and Scribe is not suitable for MANET environments. WB users perceive unacceptable high data loss and delays. Furthermore, both the Pastry overlay network and the Scribe multicast tree get frequently partitioned. This results in some WB users to be completely isolated from the rest of the network. Finally, we show that some of these problems can be avoided by using CrossROAD. Specifically, the structure of the Scribe tree is quite more stable when CrossROAD is adopted, and partitions problems experienced with Pastry completely disappear. Thus, CrossROAD turns out to be a very promising P2P system for MANET environments.

2 WB and its middleware support

The Whiteboard application was originally designed in [9], and then we adapted it to work on top of Pastry and CrossROAD. It implements a distributed whiteboard, that

*This work was partially funded by the FET-IST Programme of the European Commission, IST-2001-38113 MOBILE-MAN project.

can be used to share dynamically generated content (e.g., drawings, messages, ...). Each user runs a WB instance on her mobile device, and selects a *topic* she wants to associate to (i.e. "treasure hunting"). Each topic is linked with a canvas on which she can draw strokes or type text. On the same canvas, the user directly sees strokes and text generated by others. Being a simple example of group-communication applications, WB allows us to understand how such applications can be successfully developed in MANETs.

WB needs a subject-based multicast protocol to build groups (i.e., identify all nodes whose users are interested into the same topic), and disseminate WB data to the group members. Specifically, in our testbed, Scribe [4] is used as the multicast protocol, since it has shown to outperform other similar solutions [5]. Scribe is designed to work on top of Pastry, but it can be used with any P2P system providing a commonAPI-compliant overlay network, such as CrossROAD.

2.1 Pastry and CrossROAD

Pastry is a P2P system based on a DHT to build a structured overlay network (*ring*) at the middleware level. A logical identifier (node id) is assigned to each node hashing one of its physical identifiers (e.g., IP address, hostname). Messages are sent on the overlay by specifying a destination key k belonging to the logical identifiers' space. Pastry routes these messages to the node whose id is numerically closest to k value. To route messages, Pastry nodes maintain a limited subset of other nodes' logical ids in their internal data structures (middleware routing tables). Periodic data exchange between nodes of the overlay are needed to update the state of the overlay. Finally, in order to initially join the overlay network, each Pastry node executes a bootstrap procedure, during which it initialises its middleware routing table by collecting portions of other nodes' routing tables. Specifically, each nodes has to connect to an already existing Pastry node (i.e., it needs to know its IP address) in order to correctly start the bootstrap procedure.

The bootstrap phase and the periodic data exchange between nodes constitute the main network overhead of Pastry. CrossROAD, that is a Pastry-like P2P system explicitly designed for MANETs, drastically reduces the Pastry overhead by exploiting cross-layer interactions with a proactive routing protocol. Specifically, CrossROAD defines a cross-layer Service Discovery protocol in order to broadcast information about upper-layer services (e.g. Scribe) through the proactive flooding of routing packets, and to maintain an association between nodes' IP addresses and provided services. Hence, each CrossROAD node can autonomously build the overlay, by simply hashing the IP address of nodes providing the same service. In this way the overlay network related to a particular service is maintained with almost *negligible network overhead* in comparison with Pastry. Furthermore, CrossROAD i) is completely self-organising, since it does not require any bootstrap procedure, and ii) correctly manages cases of network partitioning and topology changes with the same delays of the routing protocols.

2.2 Scribe

Scribe exploits Pastry-like routing to build multicast groups. From the standpoint of the application running on Scribe, the group is identified by a *topic*. Scribe uses the hash function provided by Pastry (or CrossROAD) to generate the topic id (t_{id}) in the logical space of node ids. In order to join the Scribe tree, nodes send a `join` message on the overlay with key equal to t_{id}. This message reaches the next hop (say, N) towards the destination on the overlay network. The node originating the `join` message is enrolled as a child of N. If not already in the tree, N itself joins the tree by generating a `join` message anew. Eventually, such a message reaches the node whose id is the closest one to t_{id} and is not propagated further. This node is defined as the *root* of the Scribe tree.

Application messages are sent on the overlay with key equal to t_{id}. Hence, they reach the Scribe root, which is in charge of delivering them over the tree. To this end, it forwards the messages to its children, which further forward them to their children, and so on.

Finally, the Scribe maintenance procedure is as follows. Each parent periodically sends a `HeartBeat` message to each child[1]. If a child does not receive any message from the parent for a given time interval (20 s in the default case), it assumes that the parent has given up, and re-executes the join procedure. This simple procedure allows node to discover parent failures, and re-join the tree, if the case.

3 Experimental Environment

The experiments reported in this paper are based on a *static* MANET. This allows us to highlight limitations that originate from Pastry and Scribe design, rather than to mobility. Extending the results in the case of mobility is subject of future work.

The experiment testbed is as depicted in Figure 1. We set up an indoor MANET consisting of 8 nodes. To have an homogeneous testbed, all nodes are IBM ThinkPad R50 laptops. We use the built-in Intel PRO-Wireless 2200 802.11 card, with `ipw2200` driver (on Linux 2.6 kernel). The data rate is set to 11 Mbps. In addition the transmission power of each card has been adjusted to reproduce the topology shown in the figure and obtain a multi-hop ad hoc network. During the experiments, nodes marked A through to F participate in the overlay network, and run the WB application (they will be throughout referred to as "WB nodes").

[1] Application-level messages are used as implicit `HeartBeat`s.

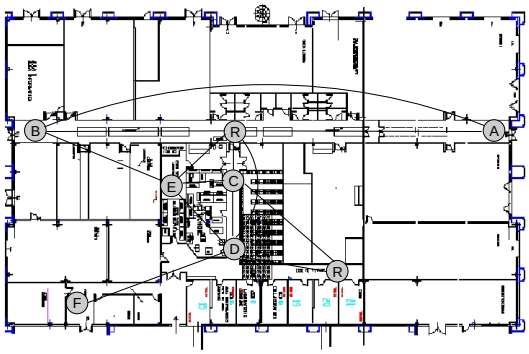


Figure 1. Map of the experiment setup

Nodes marked with "R" are used just as routers. It is worth pointing out that this setup lies within the "802.11 ad hoc horizon" envisioned in [11], i.e. 10-20 nodes, and 2-3 hops. Therefore, it is a valid example of possible real-world MANETs.

In order to have a controllable and reproducible setup, a human user at a WB node is represented by a software agent running on the node. During an experiment, each software agent interleaves active and idle phases. During an active phase, it draws a burst of strokes on the canvas, which are sent to all the other WB nodes through Scribe[2]. During an idle phase, it just receives possible strokes from other WB nodes. After completing a given number of such *cycles* (a cycle is defined as a burst of strokes followed by an idle time), each agent sends a `Close` message on the Scribe, waits for getting `Close` messages of all the other nodes, and shuts down. Burst sizes and idle phase lengths are sampled from exponentially distributed random variables. The average length of idle phases is 10 s, and is fixed through all the experiments. On the other hand, the average burst size is defined on a per-experiment basis. As a reference point, we define a traffic load of 100% as the traffic generated by a user drawing, on average, one stroke per second. Finally, the number of cycles defining the experiment duration is fixed through all the experiments. Even at the lowest traffic load taken into consideration, each agent draws – on average – at least 50 strokes during an experiment. For the performance figures defined in this paper (see below) this represents a good trade-off between the experiment duration and the result accuracy.

Some final remarks should be pointed out about the experiment start-up phase. Nodes are synchronised at the beginning of each experiment. Then, in the Pastry case, the Pastry bootstrap sequence occurs as follows[3]: node C starts first, and generates the ring. Nodes E and D start 5 seconds after C, and bootstrap from C. Node B starts 5 seconds after E and bootraps from E. Node A starts 5 seconds after B and bootraps from B. Finally, node F starts 5 seconds after D and bootraps from D. After this point in time, the Scribe tree is created and, finally, WB instances start sending application messages (herafter, WB messages). This way, the Scribe tree is built when the overlay network is already stable, and WB starts sending when the Scribe tree is completely built.

[2]Please note that in our experiment each stroke generates a new message to be distributed on the Scribe tree.

[3]The same schedule is also used to start CrossROAD, even though a CrossROAD node does *not* need to bootstrap from another node.

3.1 Performance Indices

Since Pastry and Scribe have been conceived for fixed networks, we investigate if they are able to provide an adequate Quality of Service to users in a MANET environment. To quantify the "WB user satisfaction" we use two performance indices:

Packet Loss: at each node i, we measure the number of WB messages received and sent (R_i and S_i, respectively) during an experiment; the packet loss experienced by node i is defined as $pl_i = \frac{R_i}{\sum_i S_i}$.

Delay: the time instant when each packet is sent and received is stored at the sending and receiving node, respectively. This way, we are able to evaluate the delay experienced by each node in receiving each packet. If d_{ij} is the delay experienced by node i in receiving packet j, and N_i the total number of packets received by i during an experiment, the average delay experienced by node i is defined as $D_i = \frac{\sum_j d_{ij}}{N_i}$.

Furthermore, we define two more indices, to quantify the quality of the multicast tree created by Scribe.

Node Stress: for each node, it is defined as the average number of children of that node. If t_{ij} is the time interval (within an experiment) during which node i has n_j children, the average node stress of node i is $NS_i = \frac{\sum_j n_j t_{ij}}{\sum_j t_{ij}}$.

Re-subscriptions: for each node, we count the number of times (during an experiment) this node sends new subscriptions requests, because it can't communicate with the previous parent anymore.

4 Performance with Pastry

The results we report in this section are obtained by using Pastry as DHT, and either OLSR or AODV as routing protocol. Experiments are run by increasing the traffic load starting from 20% up to 80%.

Before presenting the results in detail, let us define what herafter will be referred to as "crash of the Scribe Root Node". In our configuration Pastry assigns node ids by hashing the IP address and the port used by Scribe on the node. Hence, each node always gets the same node id. Furthermore, the topic used by the WB users is always the

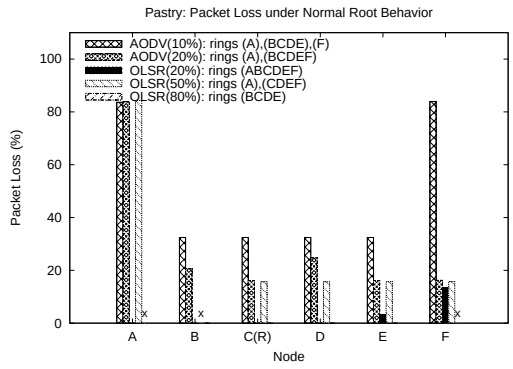


Figure 2. Packet Loss w/o MSRN crash

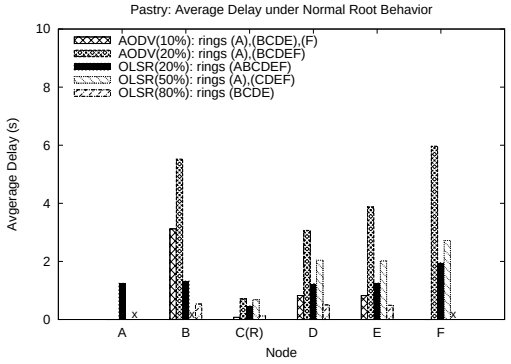


Figure 3. Delay w/o MSRN crash

same. Under the hypothesis that Pastry generates a single ring encompassing all WB nodes, the Root of the Scribe tree (i.e., the node whose id is closest to the WB topic id) is the same through all the experiments, and is node C in Figure 1. This node will be throughout referred to as the Main Scribe Root Node (MSRN). Due to the Scribe algorithm, each WB message to be distributed on the tree is firstly sent to MSRN, and then forwarded over the tree. Often, this is an excessive load for MSRN, which, after some point in time, becomes unable to deliver all the received messages. Instead, messages are dropped at the MSRN sending queue. We refer to this event as a crash of MSRN. Of course, since the application-level traffic is randomly generated, the MSRN crash is not a deterministic event.

4.1 User Satisfaction

Figures 2 and 3 show the packet loss and the delay indices experienced by the WB nodes considering experiments where the MSRN *does not* crash. Specifically, we consider AODV experiments with 10% and 20% traffic load, and OLSR experiments with 20%, 50% and 80% traffic load, respectively. There is no point in running AODV experiments with higher traffic load, since performances with AODV are quite bad, even with such a light traffic load. In the figure legend we also report the rings that Pastry builds during the bootstrap phase (please note that, theoretically, just one ring should be built, encompassing all WB nodes). Finally, an "x" label for a particular node and a particular experiment denotes that for that experiment we are not able to derive the index related to the node (for example, because some component of the stack crashed during the experiment).

Figure 2 allows us to highlight an important Pastry weakness. If a WB node is unable to successfully bootstrap, it starts a new ring, and remains isolated for the rest of the experiment. In MANET environments, links are typically unstable, and the event of a WB node failing to contact the bootstrap node is quite likely. Clearly, once a node is isolated, it is unable to receive (send) WB messages from (to) other nodes for the rest of the experiment, and this results in packet losses at all nodes. In the "AODV 10%" experiment, nodes A and F are isolated, and create their own rings. This results in packet loss of about 80% at those nodes (i.e., they just get their own WB messages, which is about one sixth of the overall WB traffic), and about 33% at nodes B, C, D and E. Similar remarks apply to the "OLSR 50%" experiment. It is more interesting to focus on the "AODV 20%" experiment. In this case, node A is isolated, while nodes B, C, D, E and F belong to the same ring. As before, A's packet loss is about 80%. The packet loss at the other nodes due to the isolation of node A is about 18% (one sixth of the overall traffic). It is interesting to notice that nodes B and D experience a *higher* packet loss, meaning that they are unable to get WB messages generated within the "main" Pastry ring (i.e., nodes B, C, D, E, F). Finally, in the case "OLSR 20%", Pastry is able to correctly generate a single ring, and the packet loss is quite low. In the case "OLSR 80%" nodes A and F crash. However, the packet loss experienced by the other nodes is negligible.

Similar observations can be drawn by focusing on the delay index (Figure 3). First of all, it should be pointed out that the delay related to nodes that are the sole member of their own ring (e.g., node A in the "AODV 10%" case) is obviously negligible. Even though – in general – the delay in this set of experiments is low, it can be noted that better performances are achieved by using OLSR instead of AODV. Finally, it should be noted that MSRN (node C) always experiences a lower delay with respect to the other nodes in the same ring.

Figures 4 and 5 show the packet loss and the delay indices in cases of MSRN crash. The packet loss experienced by nodes in the *same* ring becomes higher than in cases where MSRN does not crash. In the first three experiments, node A isolation causes a packet loss of about 18% on the other nodes. Hence, the remaining 60% packet loss is ascribed to the MSRN crash. Quite surprisingly, OLSR with 80% traffic load shows better performance than OLSR with 50% traffic load. It is also interesting to note that the packet loss at MSRN is always lower than at other nodes in the same ring. This highlights that MSRN is able to get, but un-

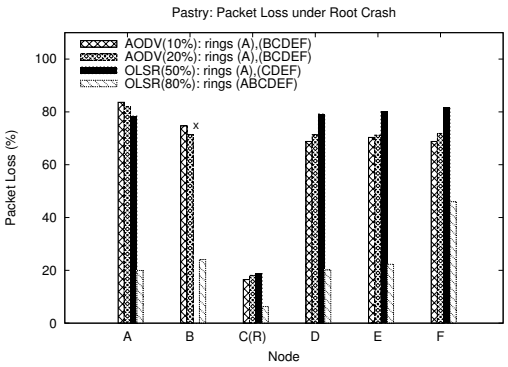


Figure 4. Packet Loss w/ MSRN crash

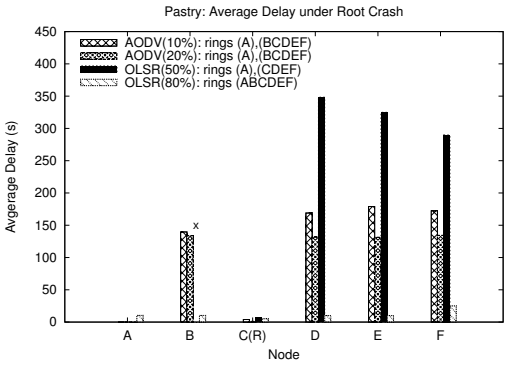


Figure 5. Delay w/ MSRN crash

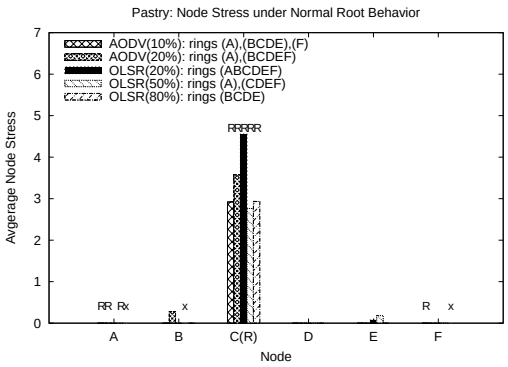


Figure 6. Node stress w/o MSRN crash

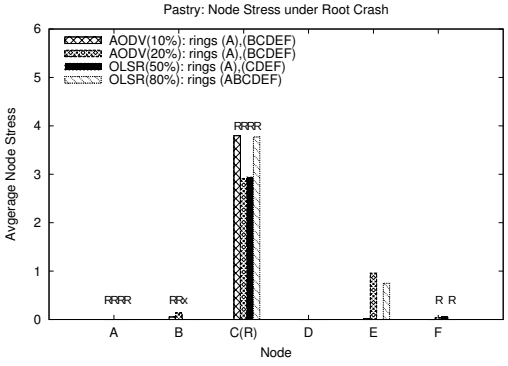


Figure 7. Node stress w/ MSRN crash

able to *deliver* over the Scribe tree WB messages generated by other nodes. Similar observations can be drawn by looking at Figure 5, as well. The delay experienced by nodes B, D, E and F can be as high as a few *minutes*, either by using AODV or OLSR. Finally, the delay experienced by MSRN is very low in comparison to the delay experienced by the other nodes.

To summarise, the above analysis allows us to draw the following observations. The Pastry bootstrap algorithm is too weak to work well in MANETs, and produces unrecoverable partitions of the overlay network. This behavior is generally exacerbated by AODV (in comparison to OLSR). Furthermore, MSRN is clearly a bottleneck for Scribe. MSRN may be unable to deliver WB messages also with moderate traffic loads, resulting in extremely high packet loss and delay. Moreover, the performance of the system in terms of packet loss and delay is unpredictable. With the same protocols and traffic load (e.g., OLSR and 50% traffic load), MSRN may crash or may not, resulting in completely different performance figures. In cases where MSRN crashes, packet loss and delay are clearly too high for WB to be actually used by real users. However, even when MSRN does not crash, the high probability of WB users to be isolated from the overlay network makes Pastry-based solutions too unreliable. These results suggest that Pastry and Scribe need to be highly improved to actually support group communication applications such as WB in MANET environments.

4.2 Multicast Tree Quality

In this section we analyse the node stress and re-subscription indices, with respect to the same experiments used in the previous section.

Figures 6 and 7 plot the average node stress with and without MSRN crashes, respectively. In both cases, the node stress is significantly higher at MSRN than at any other node. This means that the Scribe tree is a one-level tree, and MSRN is the parent of *all* the other nodes. This behavior is expected, and can be explained by recalling the way Scribe works. In our moderate-scale MANET, all nodes are in the Pastry routing table of each other. Hence, Scribe `join` messages reach MSRN as the first hop, and MSRN becomes the parent of all other nodes (in the same ring). Together with the way application-level messages are delivered, this phenomenon explains why MSRN is a bottleneck, since it has to send a distinct message to *each* child when delivering WB messages over the tree. This is a major limitation of the Scribe algorithm, and optimisations of the P2P system are clearly not sufficient to cope with it.

In Figures 6 and 7 we have added "R" labels to indicate nodes that occur to become Scribe Root during the corresponding experiment. When MSRN does not crash (Figure 6) other nodes become Scribe root only as a side effect of a failed Pastry bootstrap. On an isolated WB node,

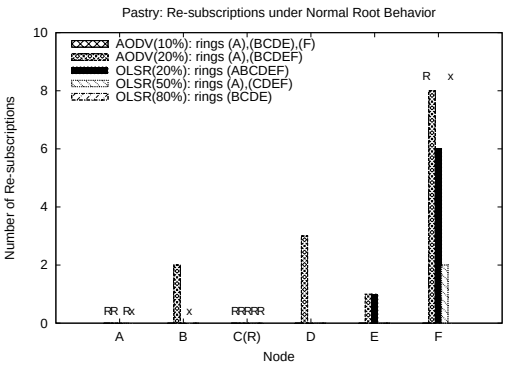


Figure 8. Re-subscriptions w/o MSRN crash

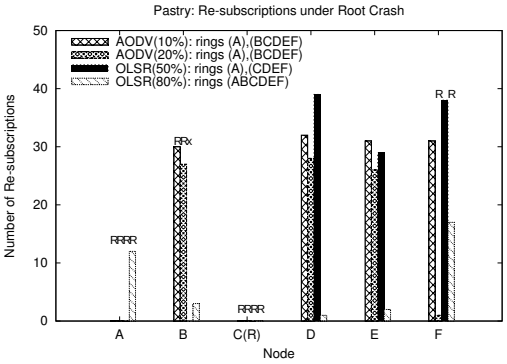


Figure 9. Re-subscriptions w/ MSRN crash

Scribe builds a tree which consists only of the node itself, that is thus the root. However, Scribe partitions may also occur due to congestion at the Pastry level in cases where MSRN crashes. By looking at Figure 7, it can be noticed that nodes other than MSRN may become root also if they belonged (after the Pastry bootstrap phase) to the same overlay network of MSRN. This phenomenon occurs, for example, at node A in the OLSR 80% case, and at node B and F (whenever they become root). It should be noted that a node with id n_1 (other than MSRN) becomes root when i) it looses its previous parent, and ii) the Pastry routing table does not contain another node id n_2 such that n_2 is closer to the WB topic id than n_1. Figure 7 shows that the congestion at the Pastry level is so high that the Pastry routing table of some nodes becomes incomplete (i.e., MSRN disappears from other nodes' routing table). Thus, the Scribe tree gets partitioned in several isolated sub-trees. Clearly, this contributes to the high packet loss measured in these experiments. Another effect of Pastry congestion during MSRN crashes is a possible reshaping of the Scribe tree. Figure 7 shows that the average Node Stress of E is close to 1 in the "AODV 20%" and "OLSR 80%" cases. This means that MSRN disappears from the Pastry routing table of some node, which – instead of becoming a new root – finds node E to be the closest one to the WB topic id. This phenomenon could be considered a benefit, since it reduces the MSRN node stress. However, it derives from an incorrect view of the network at the Pastry level, originated from congestion.

Figures 8 and 9 show the re-subscription index for the same set of experiments. Figure 8 shows that, when MSRN does not crash, the Scribe tree is quite stable. Most of the re-subscriptions occur at node F, which is the "less connected" node in the network (see Figure 1). In these experiments, the performance in the AODV cases is worse than in OLSR cases. Furthermore, upon MSRN crashes (Figure 9), the number of re-subscriptions increases drastically, even in case of "well-connected nodes" (i.e., node B, D and E). MSRN crashes make other nodes unable to get messages from their parent (i.e., MSRN itself), increasing the number of re-subscriptions. It is interesting to point out that this is a typical positive-feedback control loop: the more MSRN is congested, the more re-subscriptions are sent, the more congestion is generated.

To summarise, the multicast tree generated by Scribe on top of Pastry is quite unstable, especially in cases of MSRN crashes. The tree may get partitioned in disjoint sub-trees, and many re-subscriptions are generated by nodes. Furthermore, Scribe is not able to generate a well-balanced multicast tree, since MSRN is the parent of all other nodes. Directions to optimise Scribe are discussed in Section 6.

5 Improvements with CrossROAD

In this section we show that using a P2P system optimised for MANETs is highly beneficial to the stability of the Scribe tree. In this set of experiments, we use CrossROAD instead of Pastry, and set the traffic load to 20%, 50% and 100%, respectively. We concentrate on the performance figures related to the quality of the multicast tree, i.e., the average node stress (Figure 10) and the number of re-subscriptions (Figure 11). A complete evaluation of the User Satisfaction parameters, as well as further optimisations of the Scribe algorithm, are subjects of future work.

The first main improvement achieved by using CrossROAD is that neither the overlay network nor the Scribe tree get partitioned. CrossROAD is able to build a *single* overlay network in all the experiments. Furthermore, even at very high traffic loads (e.g., 100%), MSRN is the *only* root of the Scribe tree. Therefore, CrossROAD is able to overcome all the partition problems experienced when Pastry is used.

Figure 10 clearly shows that the node stress still remains quite unbalanced among the nodes. MSRN is typically the parent of all other nodes, and this contributes to make it a bottleneck of the system, as highlighted above. This behavior is expected, since it derives from the Scribe algorithm, and cannot be modified by changing P2P system.

Finally, Figure 11 shows that the Scribe tree is more stable (i.e., requires less re-subscriptions) using CrossROAD instead of Pastry. To be fair, we have to compare Fig-

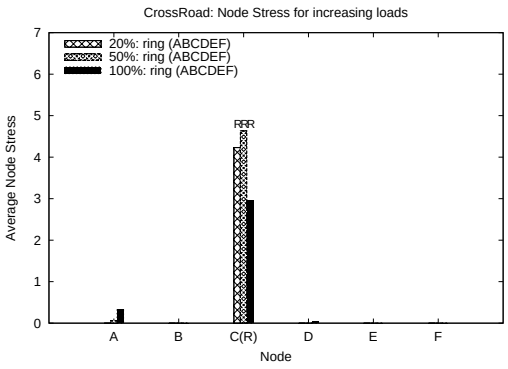


Figure 10. Node Stress with CrossROAD

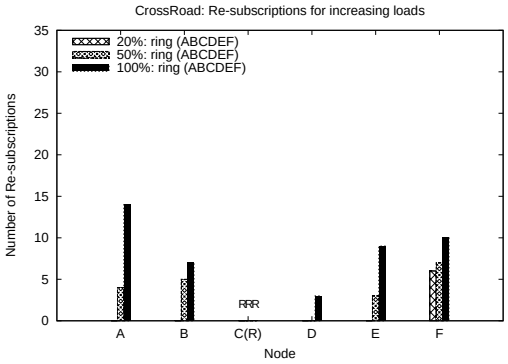


Figure 11. Re-subscriptions with CrossROAD

ure 11 with both Figures 8 and 9. It is clear that CrossROAD outperforms Pastry when used on top of AODV. The "20%" case of CrossROAD should be compared with the "OLSR 20%" case of Figure 8, since in both experiments the overlay network is made up of all nodes. The number of re-subscriptions measured at node F is the same in both cases, while it is higher at node E when Pastry is used. The CrossROAD "50%" case shows a higher number of re-subscriptions with respect to the "OLSR 50%" case in Figure 8. However, it should be noted that in the latter case the overlay network encompasses less nodes, and hence the congestion is lower. It should also be noted that, with the same nodes in the overlay network, with the same protocol stack and traffic load, Pastry experiments may suffer MSRN crashes (Figure 9). In this case, the number of re-subscriptions is much higher than in the CrossROAD case. Finally, results in the CrossROAD "100%" case should be compared with the "OLSR 80%" case of Figure 9, since the overlay network is the same in both experiments. CrossROAD achieves comparable performance, and at some nodes it outperforms Pastry, even if the application traffic is significantly higher.

5.1 Overlay management overhead

In the previous section we have shown that adopting CrossROAD significantly improves the performance of Scribe. In this section we highlight that one of the main reasons for this improvement is the big reduction of the network overhead. This is a key advantage in MANET environments.

Figure 12 shows the network load experienced by nodes A, C and by the two nodes which just act as routers, during the Pastry "OLSR 80%" experiment in which MSRN crashes[4]. Each point in the plot is computed as the aggregate throughput (in the sending and receiving directions) over the previous 5-seconds time frame. We take into consideration the traffic related to the whole network stack, from the routing up to the application layer. Specifically, nodes A and C are representative for WB nodes, pointing out the difference with nodes that just work as routers. The discrepancy between the curves related to node A and C confirms that the MSRN node has to handle a far greater amount of traffic with respect to the other WB nodes, due to the Scribe mechanisms. Furthermore, it should be noted that the curves related to the two routers can hardly been distinguished in Figure 12, since they are about $400Bps$. This means that the lion's share of the load on WB nodes is related to Pastry, Scribe and the WB application.

Figure 13 plots the same curves, but related to the "100%" CrossROAD experiment. Also in this case, MSRN (node C) is more loaded than the other WB nodes. However, by comparing Figures 13 and 12 we can highlight that the Pastry network load is far higher than the CrossROAD network load. By considering the average value over all nodes in the MANET, the Pastry load is about 3 times greater than the CrossROAD load. More specifically, the average load of C and A is 48.5 KB/s and 16.5 KB/s in the Pastry case, while drops to 21.1 KB/s and 2.96 KB/s in the CrossROAD case. The reduction of the network load achieved by CrossROAD is thus 56% at node C and 82% at node A. Since the other stack components are exactly the same, CrossROAD is responsible for this reduction[5]. Furthermore, it should be noted that, during several time intervals, the load of node A is just slightly higher than that of "routing" nodes. This suggests that the additional load of CrossROAD management with respect to the routing protocol is very limited.

6 Conclusions and Future Works

Results presented in this paper allows us to draw the following conclusions. Pastry and Scribe seem not to be good candidates to support group communication applications in MANET environments. Pastry is particularly weak during the bootstrap phase, causing the overlay network to be partitioned into several subnetworks, and some nodes to be unable to join application services. Further partitions

[4]We do not take into account AODV experiments, since OLSR has clearly shown to outperform AODV.

[5]The actual reduction is even higher, since the application-level traffic is 100% in the CrossROAD case.

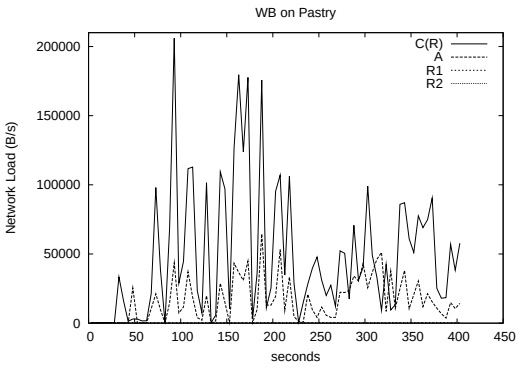


Figure 12. Network Load with Pastry

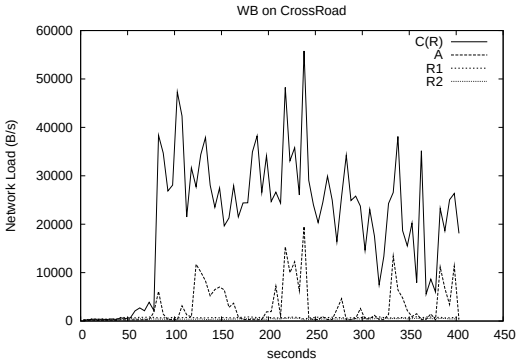


Figure 13. Network Load with CrossROAD

may occur in the Scribe tree due to congestion at the Pastry level. Finally, the delivery algorithm implemented by Scribe generates a severe bottleneck in the tree, which is highly prone to get overladed. All these limitations result in unacceptable levels of packet loss and delay for applications. Many of these problems can be avoided by adopting a cross-layer optimised P2P system such as CrossROAD. Thanks to the interactions with a proactive routing protocol CrossROAD is able to avoid all the partition problems experienced with Pastry, and to drastically reduce the network overhead. Clearly, CrossROAD cannot solve the problem of bottlenecks in the Scribe trees. Therefore, optimised versions of Scribe are required for group communication applications such as WB to be really developed in MANETs. The direction we are exploring is building a single distribution tree, optimised through cross-layer interactions with a proactive routing protocol. Building a single-tree, instead of a new tree for each source node, allows for a more scalable solution. In addition, cross-layering allow us to retain the subject-based features of Scribe (e.g., locating a tree by means of its topic), while exploiting also topological information to build the tree. Furthermore, the tree can be built in a completely distributed way, by exploiting greedy policies such as those used in YAM [3] and ALMA [10]. Finally, the data-distribution phase can be optimised so as to avoid each message to be sent to the MSRN and *then* delivered to other nodes. For example, while travelling towards MSRN, messages can be duplicated and delivered at each branching point in the tree. These policies are expected to drastically mitigate the bottleneck problems experienced by Scribe. Furthermore, they raise very interesting arguments about causal ordering of messages, that we are planning to address, as well.

References

[1] G. Anastasi, E. Borgia, M. Conti, E. Gregori and A. Passarella, “Understanding the Real Behavior of Mote and 802.11 Ad hoc Networks: an Experimental Approach”, Pervasive and Mobile Computing, in press.

[2] AODV, Dept. of Information technology at Uppsala University (Sweden), http://user.it.uu.se/ henrikl/aodv/.

[3] K. Carlber and J. Crowcroft, “Building Shared Trees Using a One-to-Many Joining Mechanism”, ACM Computer Communication Review, pp. 5-11, Jan. 1997.

[4] M. Castro, P. Druschel, A-M. Kermarrec and A. Rowstron, “SCRIBE: A large-scale and decentralised application-level multicast infrastructure”, IEEE Journal on Selected Areas in Communication (JSAC), Vol. 20, No, 8, October 2002.

[5] M. Castro, M. B. Jones, A-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang and A. Wolman, “An Evaluation of Scalable Application-level Multicast Built Using Peer-to-peer overlays”, Infocom 2003, San Francisco, CA, April, 2003.

[6] F. Dabek and B. Zhao and P. Druschel and J. Kubiatowicz and I. Stoica, “Towards a common API for Structured Peer-to-Peer Overlays”, Proc. of the the 2nd International Workshop on Peer-to-peer Systems (IPTPS’03), Berkeley, CA, Feb. 2003.

[7] M. Conti, G. Maselli, G. Turi, and S. Giordano, “Cross layering in mobile ad hoc network design”, IEEE Computer, Feb. 2004.

[8] F. Delmastro, “From Pastry to CrossROAD: Cross-layer Ring Overlay for Ad hoc networks”, in Proc. of Workshop of Mobile Peer-to-Peer 2005, in conjuction with the PerCom 2005 conference, Kauai Island, Hawaii, Mar. 2005.

[9] M. Dischinger, “A flexible and scalable peer-to-peer multicast application using Bamboo”, Report of the University of Cambridge Computer Laboratory, 2004, available at http://www.cl.cam.ac.uk/Research/SRG/netos/futuregrid/dischinger-report.pdf.

[10] M. Ge, S.V. Krishnamurthy, and M. Faloutsos, “Overlay Multicasting for Ad Hoc Networks”, Proc. of the Third Annual Mediterranean Ad Hoc Networking Workshop (MedHocNet 2004), June 2004.

[11] P. Gunningberg and H. Lundgren and E. Nordström and C. Tschudin, “Lessons from Experimental MANET Research”, Ad Hoc Networks Journal, (Special Issue on “Ad Hoc Networking for Pervasive Systems”), Vol. 3, Number 2, March 2005.

[12] OLSR, Andreas Tonnesen, Institute for informatics at the University of Oslo (Norway), http://www.olsr.org.

[13] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems”, Middleware 2001, Germany, November 2001.

A Comparative Study of Cooperative Algorithms for Wireless Ad Hoc Networks

Alan Lim, Vikram Srinivasan, Chen-Khong Tham
Department of Electrical and Computer Engineering
National University of Singapore
Engineering Drive 3 Singapore 117576
{eng10346, elevs, eletck}@nus.edu.sg

Abstract

In Ad-Hoc Networks (AHN), nodes can take on the role of a source, destination or relay. Relays are needed when source nodes wish to communicate with destination nodes that are far away. However, in acting as relays, nodes expend a significant amount of energy transmitting data across the network. Energy is a precious commodity for an AHN and the aim of nodes is to maximize their throughput and survive for the longest period possible. Thus, nodes would not be willing to act as relays all the time. As a result, efficient cooperative algorithms are needed in AHNs to determine when a node should act as a relay, thus ensuring the survivability and performance of nodes. However, there are issues that need to be considered before existing cooperative algorithms may be implemented in an actual AHN. This paper focuses on the research, modification, implementation, verification and analysis of the Generous Tit for Tat (GTFT) and Nuglet cooperative routing algorithms on an actual AHN.

1. Introduction

Ad-hoc Networks (AHNs) have vast potential applications in the modern world. These include situations that demand the rapid deployment of nodes. AHNs can also extend the coverage of a network significantly without incurring significant financial costs. A node in an AHN can typically take the form of a PDA or a laptop. AHNs usually have an arbitrary and random configuration. In fact, the number of nodes that come together to form a network at any instance is non-deterministic. Furthermore, we assume that nodes have an equal chance at any one time of being a source, destination or relay.

A node in an AHN is faced with two primary constraints. Firstly, in the transmission of data packets, energy (in terms of the battery levels of the nodes) is consumed. Thus, since PDAs or laptops are portable systems allowing users to process information on the go, they are heavily dependant on the limited battery power that they carry. Consequently, nodes would want to conserve as much energy as possible. Secondly, nodes would also like to have the maximum goodput (number of packets accepted by the relays over the number of packets sent out as a source) possible. However this would require relay nodes to cooperate all the time. While it may be intuitive that relay nodes should help to forward packets for other nodes all the time, it is not in their interest to do so. If a relay node were to transmit data continuously for other nodes, there may be little or no energy left for its own use.

References such as [1], [2], [3], [4] and [5] propose the use of cooperative algorithms to determine if a node should forward a data packet for another node. The primary aims of these algorithms are to maximize the throughput and lifetime of a node. Other approaches like [6], [7] and [8] propose mechanisms to be employed in the event of misbehaving nodes. Meanwhile, references [9] and [10] have studied the energy consumption of wireless transmission.

This paper uses some of these ideas with slight modifications to the theoretical framework where necessary. While previous works have been supported purely by simulation results, this is, to the best of the authors' knowledge, the first paper that looks at the implementation issues of cooperative algorithms on an actual AHN. Specifically, the GTFT [1] and Nuglet [2] cooperative algorithms were implemented. In this paper we study the results obtained and verify them against existing simulation results. We also compare the algorithms against each other and with the case when no cooperative algorithm was used.

In section 2, the algorithms employed in [1] and [2] are briefly discussed. Next, in section 3, the setup of the experiments is given. In section 4, the necessary modifications to the existing Nuglet and GTFT algorithms are discussed in detail and their implementation is explained. After that, in section 5 the experimental settings are given, while the experiment results are shown and analyzed in section 6. Finally, in section 7, the paper is concluded.

2. Related Work

2.1. Nuglet Algorithm

The Nuglet algorithm, described in [2] seek to stimulate packet forwarding by rewarding nodes who participate in packet forwarding. It does this through the use of a nuglet counter, which is incremented whenever nodes forward packets for others. Conversely, the counter is decremented by an integer constant *N* whenever a node acting as a source, sends out a packet. Nodes are not allowed to send out packets if they do not have sufficient counters. Hence, nodes are encouraged to forward packets in order to increase their counter values, thereby enabling them to send out data packets subsequently. Besides the nuglet counter, the algorithm also includes a battery counter. The battery counter is decremented whenever a node sends or receives a data packet. It represents the number of packets that a node can send out before its battery runs out. In [2], the routing decisions of a node is found to be constrained by the following equations.

$$Out_r, Out_s \geq 0 \tag{1}$$

$$N^* Out_s - Out_r \leq C \tag{2}$$

$$Out_s + Out_r = B \tag{3}$$

$Where:$

Out_s	$: Number\ of\ packets\ sent\ out\ by\ the\ node$
Out_r	$: Number\ of\ packets\ relayed\ by\ the\ node$
N	$: Integer\ constant$
C	$: Nuglet\ counter$
B	$: Battery\ counter$

From equations (1) - (3), Fig 1. is obtained. The optimum operating point for the nodes corresponds to the intersection between the two lines in Fig 1. The maximum number of output and forwarded packets in equations (4) and (5) are then obtained from the respective Out_s and Out_r values at the intersection point. When a relay has forwarded as many packets as the maximum number of output packets, further relay requests will not be accepted. This is because its not in it's interest to forward any more packets, as it has acquired enough nuglets to send out as many packets as a source to drain out its battery counter.

$$\text{Maximum no. of output packets} = \frac{B+C}{N+1} \tag{4}$$

$$\text{Maximum no. of forwarded packets} = \frac{NB-C}{N+1} \tag{5}$$

While the Nuglet algorithm performs packet based processing like other micro payment schemes. It does not require any additional infrastructure (such as the accounting center or base stations described in [4]) that makes the realisation of an AHN tricky. Furthermore, unlike other similar micro payment schemes such as [4], it does not involve additional information such as the ticket data in every packet. Thus with a smaller packet size, fewer network congestions are expected. Bearing these in mind, it has the potential to serve as a simple and efficient solution.

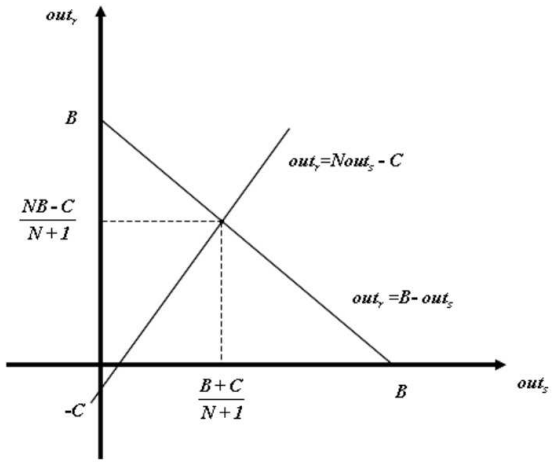


Figure 1. Maximisation of Source Output under Nuglet Algorithm from [2]

The work on the Nuglet algorithm in [2] focuses primarily on how nodes can be stimulated to forward packets. It assumes that without cooperative algorithms in place, none of the nodes will be inclined to forward packets. It shows how the throughput of the nodes can be improved, when nodes are stimulated to forward packets with the expectation that other nodes will return the favor. The paper however does not address the effects of the algorithm on the overall performance of the network in terms of energy consumption and potential energy savings. There is no comparison made between the network before and after the employment of the Nuglet algorithm.

2.2. GTFT Algorithm

The GTFT algorithm is based on game theory as described in economics. Unlike other routing algorithms including the Nuglet algorithm, nodes under GTFT are not rewarded with virtual currency whenever they forward a data packet. Instead, they are encouraged to forward a data packet because they expect other nodes to react negatively and drop their own requests in the future if they do not cooperate by forwarding packets as often as they should.

Another difference is that GTFT is a session based routing algorithm. Before a source node sends out any data packets, it will first send out a session request. Relays then decide whether to accept all relay requests for the entire session or none at all. Nodes refuse session requests either if it

has forwarded more sessions than the Pareto optimal value or if the rate at which it is accepting sessions (less the generosity value, ε) is greater than the rate at which other nodes are accepting its session requests. In other words, if the generosity value is increased, nodes are more willing to accept session requests even if they have not received an equivalent amount of help.

Next, nodes are classified according to their power constraints. Power constraint is defined to be the ratio between the initial energy allocated to the node and its expected lifetime. The corresponding power constraint associated with the current session can then be derived from the node with the lowest power constraint. This is because there is no incentive for a node with a higher power constraint to behave more liberally in forwarding packets since it knows that other nodes with lower power constraints are not able to reciprocate in kind.

For a particular session, a node can be a source, relay or destination. Energy is spent by a node both as a source as well as a relay for others. In fact, the total amount of energy that is spent as a source as well as a relay will be constrained by its power constraint. This is illustrated in (6).

$$\sum_{j=1}^{k} e_{pj}^{s} + e_{pj}^{r} \leq p_{class(p)} \tag{6}$$

$Where:$

$e_j^s:$ $Average\ energy\ spent\ per\ slot\ as\ a\ source$

$e_j^r:$ $Average\ energy\ spent\ per\ slot\ as\ a\ relay$

$K:$ $Total\ number\ of\ sessions$

$j:$ $Current\ session$

$p_{class(p)}:$ $Power\ constraint\ of\ particular\ class\ p$

GTFT also has the feature of a dynamic algorithm. The forwarding rate is not only dependent on the initial power settings but on the prevailing network condition as well. If the current rates at which other nodes are accepting relay requests are less than what the node itself is accepting for instance, the node will adapt accordingly and forward less packets for others in the future.

In light of the features described, GTFT has the potential to serve as an efficient solution just like the Nuglet algorithm. However, while the Nuglet algorithm adopts a packet based approach, GTFT performs its routing decision at a session level. The dynamic nature of GTFT also contrasts with the relatively static Nuglet algorithm. The two algorithms are thus chosen for study because they represent two plausible yet different solutions.

3. Experimental Setup

This section details the setup of the nodes used in the experiments. Details of the physical setup is given in Table 1. A single relay setup proved to be adequate in capturing the salient points of our research without involving a large number of operators and nodes. The iPAQ 5500 came with its own wireless LAN support while the Cisco Aironet 350 wireless card were used by the laptop and the iPAQ 3630.

Table 1. System Specifications

Description	
Hardware	1 Toshiba Tecra 9100 laptop 1 Compaq iPAQ 3630 1 HP iPAQ 5500 Pocket PC 802.11b Cisco Aironet 350 wireless card
Operating System	iPAQs: Familiar project Linux v0.7.2 distribution with a 2.4.29-rmk6-pxa1-hh30 kernel Laptop: Redhat 9.0 with a 2.4.20-8 kernel
Topology	3 node string topology

3.1. Packet Forwarding

The physical implementation of an actual AHN can prove to be fairly difficult. Firstly, nodes have to be placed a significant distance apart in order for a hop to occur. Secondly, the high mobility of nodes in an AHN means that nodes have to constantly change their positions with reference to each other. These features means that a large physical space is needed to accommodate all the nodes. Furthermore, operators might be needed at each node to change the geographical positions of the nodes during the experiment.

To overcome the first constraint, we modified the way routing tables were established. Specifically, the changes were made such that end nodes can only accept route replies from relay nodes and not from each other during route discovery. Unlike end nodes, the routing table of the relay node remains unchanged, and it was able to receive route replies from all it's neighbors during route discovery. These changes allowed all the nodes to be placed next to each other during the experiment and ensured that all packet transfers went through an intermediate node.

To overcome the second constraint, we had a command center that randomly selected nodes at the start of each session to take on either the roles of a source, relay or destination. For a 3 node topology, there were 6 possible configuration that resulted from this selection. After the roles were defined, the information was broadcasted to the nodes. Nodes then configured their routing tables based on this information. After the nodes have been configured, the source node proceeded to send out the data packets to the destination node. At the end of the session, the command center was notified so that initialization and configuration was carried out for the next test case.

3.2. Measuring Energy Consumption

In [9] and [10], energy consumption was measured directly from the voltage and current drawn by the network interface card (NIC). This required an external piece of hardware such as the Sycard CardBus extender that allowed the NIC to be inserted into it. The energy measurements were then taken via a digital oscilloscope and multimeter to the terminals of the extender.

It is not in line with the objectives of this paper to model the energy consumption of data packets. Instead, we applied the average current and voltage measurement in [10], to obtain the average energy consumed in sending, relaying and receiving packets. The power constraint and initial battery counter values in our experiments were obtained this way as well.

This approach has several advantages. Firstly, nodes could be plugged into a power supply permanently during the course of the experiment instead of running on batteries. Without recharging the batteries after every experiment, a considerable amount of time is saved. Next, a fairer measurement is achieved as the amount of energy consumed in transmitting data packets is constant in all experiments. Thus, the difference in results obtained in the experiment is purely due to the routing algorithms and not the result of any inconsistency that may arise during the course of measuring the energy consumption physically. Finally, the results obtained can be easily applied to other nodes with different power specifications by substituting their individual specifications into the equations.

4. Modifications made to existing algorithms

4.1. Nuglet Algorithm

In [2], the energy consumed in sending out packets as a source and as a relay is assumed to be constant and equal to 1. Thus, the battery counter is chosen to be equal to the sum of the number of forwarded and generated packets.

4.1.1 Inclusion of packet received term

However, papers such as [9] and [10] indicates that the amount of energy consumed in receiving packets is significant and cannot be ignored. Thus, the term Out_d should be added to the equations. Out_d here refers to the number of packets that a node would like to receive as a destination.

$$Out_d, Out_s, Out_r \geq 0 \tag{7}$$

$$N^* Out_s - Out_r \leq C \tag{8}$$

Equation (7) involves an additional Out_d term while Equation (8) is the same as the original equation in [2]. The term Out_d has no effect on the nuglet counter here as the cost of transmitting the packet had already been paid by the source node. Thus, the only effect of receiving a data packet is to decrement the battery counter. Equation (9) shows the maximum possible energy consumption of the node from the individual roles given the limited energy available.

$$\begin{aligned} &\text{Maximum possible power consumption} \\ &= E_{recv}(Out_d) + E_{source}(Out_s) + E_{relay}(Out_r) \end{aligned} \tag{9}$$

$$\begin{aligned} &Where: \\ &Erecv \quad : Energy\ consumed\ in\ receiving\ a\ data\ packet \\ &Erelay \quad : Energy\ consumed\ in\ relaying\ a\ data\ packet \\ &Esource \quad : Energy\ consumed\ in\ sending\ a\ data\ packet \end{aligned}$$

4.1.2 Difference in energy consumption

Although the energy consumption of nodes varies according to roles in the real system, the difference between the energy consumed as a source and relay is minimal. They are therefore treated equally here. The energy consumed as a destination and the energy consumed as a source or relay can be expressed as a ratio. This simplifies calculations later in the paper.

$$r = \frac{Energy\ consumed\ as\ a\ destination}{Energy\ consumed\ as\ a\ source/relay} \tag{10}$$

In [2], the value of the battery counter *B*, is equal to the number of packets that a node can send out as a source. Similarly here, the value of *B* can be obtained by dividing the total energy available by E_{source}. This gives the maximum number of packets that a node can send out as a source. This is shown in Equation (10).

$$\begin{aligned} &\text{Maximum possible power consumption} \\ &= E_{recv}(Out_d) + E_{source}(Out_s) + E_{relay}(Out_r) \\ &= E_{recv}(Out_d) + E_{source}(Out_s + Out_r) \\ &= E_{source}(r^* Out_d + Out_s + Out_r) \\ \\ &\text{Total no. of packets it is able to send out as a source} \\ &= B = (r^* Out_d + Out_s + Out_r) \end{aligned} \tag{11}$$

Using the equations (7), (8) and (11), we plot Fig. 2. The optimum operating point is again given to be the intersection between the two lines. Based on this graph, we obtain the maximum number of output packets and forwarded packets as follows:

$$\begin{aligned} &\text{Maximum no. of output packets} \\ &= \frac{B - r^* Out_d + C}{N+1} \end{aligned} \tag{12}$$

$$\begin{aligned} &\text{Maximum no. of forwarded packets} \\ &= \frac{N(B - r^* Out_d) - C}{N+1} \end{aligned} \tag{13}$$

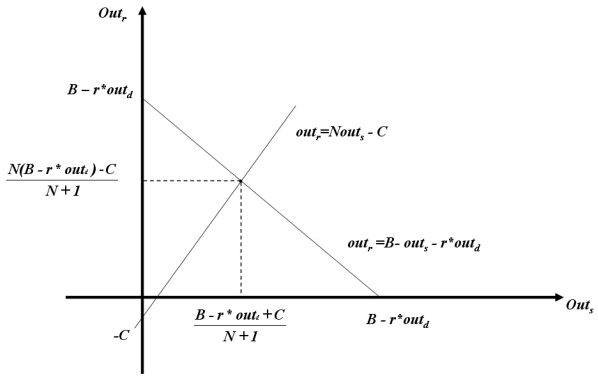


Figure 2. Modified maximization of output packets graph under Nuglet algorithm

4.2. GTFT Algorithm

In [1], the energy consumed as a destination is also assumed to be negligible and the energy required in relaying a session is taken to be constant and equal to 1 for all nodes. However, as in the Nuglet scheme, modifications has to be made to the existing equations.

4.2.1 Additional Destination Term

The energy consumed in receiving packets needs to be considered and the modified power constraint equation with the additional receiving energy term should then be

$$\sum_{j=1}^{k} e_{pj}^{s} + e_{pj}^{d} + e_{pj}^{r} \leq p_{class(p)} \qquad (14)$$

$Where:$

$e_j^s:$ $Average\ energy\ spent\ per\ slot\ as\ a\ source$

$e_j^r:$ $Average\ energy\ spent\ per\ slot\ as\ a\ relay$

$e_j^d:$ $Average\ energy\ spent\ per\ slot\ as\ a\ destination$

$K:$ $Total\ number\ of\ sessions$

$j:$ $Current\ session$

$p_{class(p)}:$ $Power\ constraint\ of\ particular\ class\ p$

4.2.2 Factoring the Average Energy Consumed

Next, the energy consumed in data transmission depends on the hardware architecture of the nodes and is not equal to 1. Factoring in the difference in energy consumed per packet for different roles affects the way energy classes are determined. Unlike in [1] where energy classes are determined simply by looking at the power constraint of the nodes, there is a need to consider the average energy consumed by the node in each role as well. iPAQs, for example, are known to be more energy efficient. For each packet transmission, less energy is consumed compared to that for a laptop. Thus, instead of looking at the power constraint, we associate the energy class with τ, the probability of a node accepting a relay request. A higher probability of accepting a relay request means a higher energy class.

Table 2. Experimental Settings

Setting	Value/Detail
Number of nodes	3
Number of test cases	100
Size of payload (bytes)	1,448
Number of packets per test case	1,000
Number of sessions per test case	1
Configuration time interval (s)	10
Routing algorithm	AODV
Packet generation rate	random

5. Experimental Settings

In Table 2, the experiment settings are outlined. A configuration time interval was introduced between sessions because nodes need time after each session to prepare their routing table for the next test case. The AODV routing algorithm is chosen for its ability to perform well under situations which demand a higher level of mobility [11].

The cooperative algorithms tested here were deployed with slight modifications to the kernel module developed by NIST [12]. Besides the cooperative algorithms mentioned earlier, experiments were also conducted using the original kernel module without modifications. We will refer to this in our discussion as the *Original* AODV scheme.

In [2], the nodes are assumed to be generating packets at a fixed rate. We have however, taken a different approach in our experiments. In our experiments, we assumed that a single session was carried out for each test case. Before the start of each test case, the nodes were randomly chosen to assume the role of a source, relay or destination. The rates at which the individual nodes were generating packets was thus random and dependent on the frequency it was chosen to be a source. This approach adds realism to the experiments, as nodes in an actual network do not have a fixed role, and their roles are expected to change over time.

The measurement of energy consumption was restricted to the duration of the session as the energy consumed during the configuration time interval is inconsequential. In reality, the roles that nodes plays in a particular session are defined by the nodes themselves instead of an external command center. The concept of a configuration time interval then, would be irrelevant.

Finally, we assumed that nodes in the AHN are not malicious and do not misbehave. Consequently, we do not consider measures such as the watchdog and pathrater in [6],

CONFIDANT in [7] and CORE in [8].

6. Experiment Results

6.1. Comparison between Routing Algorithms

In an AHN, the performance of a node is affected by other nodes. Hence, improving the performance of one node could be at the expense of others. However, as long as the overall performance of the network has improved, the optimisation objectives have been met. Here, the respective algorithms are compared against several performance indicators.

6.1.1 Relay Ratio

Ideally, a node would want as low a relay ratio (proportion of packets that are relayed for others as compared to the total number of incoming and outgoing packets) as possible. In the experiment, we observe that while the relay ratio has improved for some of the nodes under the GTFT scheme, this ratio has deteriorated for the other nodes. However, when the Nuglet scheme was adopted, a relay ratio of 33% was achieved by all nodes. This suggests that the Nuglet algorithm would lead to a more equitable forwarding burden between nodes. The unequal distribution under Original AODV and GTFT is attributed to the different packet generation rate among nodes. In our experiment, the rate at which individual nodes were chosen to be a source node was random and thus the number of relay requests varied between nodes. This is a realistic view of how the network would function in the short run. For a particular time interval, it is reasonable to expect that some nodes will have more requests than others. However in the long run, we can expect all nodes to have the same number of relay requests and the disparity described earlier would no longer by relevant. Thus, the advantage that the Nuglet algorithm has in ensuring that the forwarding burden is evenly distributed might not be as evident in the long run.

6.1.2 Portion of Energy Spent As a Relay for Others

Earlier we mentioned that nodes can take on either the role of a source, relay or destination. Energy is consumed in the respective roles, but we are interested to find out the portion of energy that is spent relaying. Ideally, nodes wish to spend as little energy relaying as possible so that more energy may be available to carry out activities that benefit them directly, such as sending and receiving packets. In this respect, the GTFT cooperative algorithm is seen to outperform the other schemes. Referring to Fig 3, the portion of total energy that nodes spent as a relay under GTFT as a network was the least. Under GTFT, 26.3% of the energy spent as a network

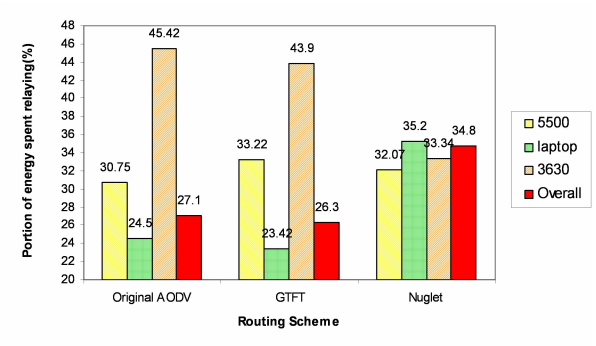


Figure 3. Portion of energy spent relaying under the different schemes for the individual nodes and the overall AHN.

was in relaying packets for others. This is an improvement from the 27.1% achieved when Original AODV was employed. The Nuglet algorithm achieved 34.8%, which is worse than if no cooperative algorithm was used (i.e when Original AODV was used). However, we find that under the Nuglet algorithm, each of the nodes spent the same portion of energy relaying (≈33%), unlike the other schemes. This again suggests that the Nuglet algorithm distributes the burden of forwarding packets equally between the nodes.

6.1.3 Average Energy Cost per Packet Sent

The amount of energy needed to send a data packet under GTFT is similar to that for Original AODV. On the other hand, the amount of energy required by the Nuglet scheme to send out the same data packet is considerably larger. The reason for this is that packet drops occur frequently at the relay nodes using the Nuglet scheme. Under GTFT, a session request instead of a packet request is sent to the relay node at the start of a session. If the request has been denied, no data packets are sent out to the relay nodes during the duration of the session. The command center is then notified so that configuration for the next session is started. This way, unlike the Nuglet scheme, energy is not wasted when data packets are sent to the relay nodes and dropped subsequently. A session request message has a smaller payload than the data packet itself. It is therefore assumed to be negligible and not included in the energy measurement. This explains why the energy cost of sending out a data packet for both GTFT and Original AODV is similar. Hence, GTFT has an advantage over the Nuglet algorithm in that the amount of energy needed to send a packet of data is relatively unchanged from Original AODV, and less than when the Nuglet algorithm was employed.

6.1.4 Number of Packets Sent During Lifetime

Ideally, nodes should have a lifetime long enough for it to send out all its data packets. As a network, this comes up to 100,000 data packets. However, given the limited battery power, this is not possible.

Using Original AODV, a total of 85,443 packets were sent out before the battery lifetime expired. Meanwhile under GTFT, 85,000 data packets were sent. The slight difference is attributed to the type of sessions that were encountered by both algorithms before their battery ran out. Under GTFT, by denying session requests occasionally, energy is conserved, and more test cases can be run. Thus, nodes under GTFT could possibly accept session requests in a later test case, that nodes under Original AODV would not have heard of. The nodes in these later test cases could be arranged in a different configuration from the one experienced by Original AODV. For example, nodes under Original AODV might have accepted 10 sessions with the laptop as the relay and 9 sessions with the iPAQs as the relay, while nodes under GTFT might have accepted 9 sessions with the laptop as the relay and 10 sessions with the iPAQs as the relay. Although the number of sessions accepted by both algorithms are the same, the energy consumed in accepting the sessions differs. For instance, the energy consumed in sending out packets from the laptop is more than that from the iPAQ. As a result of this disparity, the number of packets that nodes are able to send out during their lifetime will differ as well.

Next, we consider the Nuglet scheme with a nuglet counter value of 1000. Under this scheme, only 18,008 packets were sent out. This corresponds to about 21% of the packets sent out when GTFT or Original AODV was used. Under the Nuglet algorithm, once nodes have acquired the nuglets necessary to send out the maximum amount of packets, packet forwarding ceases. Consequently, in the later test cases, although all the nodes have acquired the necessary nuglets to send out the remaining packets, none of the nodes are able to send out anything. This is because nodes are unable to find intermediate nodes that are willing to forward their packets to the destination node. Any future requests from the source nodes will be denied. Nodes wait expectantly, hoping to trade in the nuglet counters that they have acquired but are unsuccessful. Energy will then be wasted transmitting packets only to have them dropped at the relays.

6.2. GTFT Study

Besides comparing the cooperative algorithms, we also seek to verify the simulation results of the GTFT algorithm in [1] against an actual AHN.

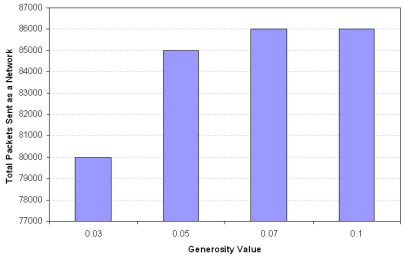


Figure 4. Total Packets Sent as a Network against Generosity Value

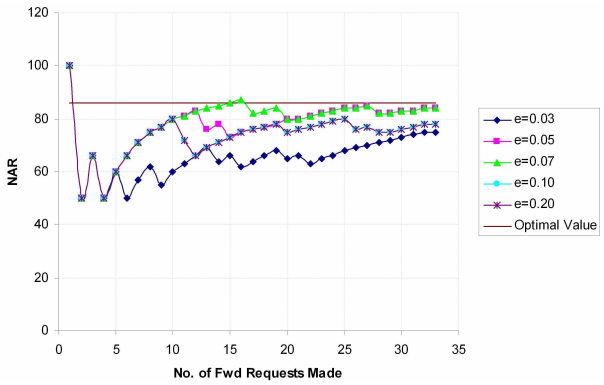


Figure 5. Convergence of NAR value

6.2.1 Increase in Packets Sent as a Network

Here, we look at how the number of packets sent by the individual nodes is affected by the generosity value. Referring to Fig. 4, we see that by increasing the generosity value, the total amount of packets that were sent increased considerably. This is as expected, as the more generous nodes are, the more sessions they are willing to forward, leading to an increase in the number of packets that are sent.

However, there is a limit to how much the nodes will send out as a network. When the generosity value was increased from ε=0.03 to ε=0.10, we found that the energy limitations were exceeded when the generosity value was set to be greater than 0.05. In addition, when energy limitations were disregarded and the generosity value was increased from 0.07 to 0.10, there was no apparent increase in the total number of packets that were sent out. This suggests that there is no incentive for nodes to be extra generous.

6.2.2 Convergence of Normalised Acceptance Rate

The simulation results in [1] shows that if all the nodes in the AHN employ GTFT, the Normalised Acceptance Rate (NAR) values will converge to the Pareto optimal value. NAR is defined to be the ratio between the number of successful relay requests and the number of relay requests that

Table 3. Summary of Algorithm Comparison

Feature	Original	Nuglet	GTFT
Less energy spent as a relay	x	x	✓
Low energy cost per packet	✓	x	✓
Large no. of packets sent during lifetime	✓	x	✓
Equitable forwarding burden	x	✓	x

x Performs less satisfactorily compared to the best scheme
✓ Performs the best compared to the other schemes

have been made by the node. The Pareto optimal value is the ideal NAR given the current network configuration. The results obtained here shows the theoretical convergence to be true (refer to Fig 5). In addition, we find that when the generosity value is increased, the NAR converges faster towards the Pareto optimal point. However, when nodes reach the optimal point they will be less willing to accept session requests. If a node has not attained the optimal value at this point, it may never be able to do so. This is because other nodes seeing their session requests rejected will reciprocate by refusing session requests. On the other hand, if the generosity value is too low, the NAR is found to be far from the Pareto optimal point although it still converges towards it. This could be because its convergence rate is so slow that it is not able to converge to the Pareto optimal value within the time frame of the experiment.

7. Conclusion

In summary, the theoretical framework and simulation results of cooperative algorithms suggested in theoretical papers were compared and verified against an actual AHN. A feasibility study was also conducted to compare the merits and drawbacks of each scheme. From Table 3, we see that GTFT gives the best performance. It reduces the energy that nodes spend relaying and allows them to send out a large number of packets during their lifetime. While the Nuglet scheme is not as favorable according to most of the performance indicators here, it is able to distribute the forwarding burden among nodes for experiments conducted in the short run. The current work assumed a single relay setup, a single session per test case and that nodes do not misbehave. Our future work would involve observing how the AHN performs after extending the current setup to a multiple relay network and allowing multiple concurrent sessions. Finally, we seek to study the effects of incorporating the measures in [6], [7] and [8] when there are misbehaving nodes.

References

[1] V. Srinivasan, P. Nuggehalli, C.F. Chiasserini and R.R. Rao, "Cooperation in Wireless Ad Hoc Networks", *Proc. of IEEE INFOCOM 2003*, San Francisco, April 2003

[2] L. Buttyan and J.P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Network", *Technical Report No. DSC/2001/046*, August 2001

[3] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J.P. Hubaux, and J.Y. Le Boudec, "Self-Organization in Mobile Ad-hoc Networks: the Approach of Terminodes", *IEEE Communications Magazine*, Vol.39 No.6, June 2001

[4] M. Jakobsson, J.P. Hubaux and L. Buttyan, "A Micropayment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks", *Proc. of Financial Crypto 2003*, La Guadeloupe, January 2003

[5] L. Buttyan and J.P. Hubaux, "Enforcing Service Availability in Mobile Ad-Hoc WANs", *Proc. of IEEE/ACM Workshop on Mobile Ad-hoc networking and Computing (MobiHOC)*, Boston, USA, August 2000

[6] S. Marti, T.J. Giuli, K. Lai and M. Baker, "Mitigating Routing Misbehviour in Mobile Ad-hoc networks", *Proc. Of Mobicom 2000*, Boston, USA, August 2000

[7] S. Buchegger and J.-Y. Le Boudec, "Performance Analysis of the CONFIDANT Protocol (Cooperation of Nodes: Fairness in Dynamic Ad-hoc NeTworks)", *Proc. Of Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, Switzerland, June 2002

[8] P. Michiardi and R. Molva, "CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad hoc Networks", *Communication and Multimedia Security 2002*, Portoroz, Slovenia, September, 2002

[9] L.M. Feeney and M. Nilsson, "Investigating The Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment", *Proc. of INFOCOM 2001*, Anchorage, Alaska, USA, August 2000

[10] B. Wang and S. Singh, "Computational Energy Cost Of TCP", *Proc. of INFOCOM 2004*, Hong Kong, March 2004

[11] A. Jain, A. Pruthi, R.C. Thakur and M.P.S. Bhatia, "TCP Analysis Over Wireless Mobile Ad Hoc Networks", *Proc. of 2002 IEEE International Conference on Personal Wireless Communicatons* , New Delhi, India, December 2002

[12] National Institute of Standards and Technology, "Kernel AODV", http://www.antd.nist.gov/wctg/aodv_kernel

A Path Density Protocol for MANETs

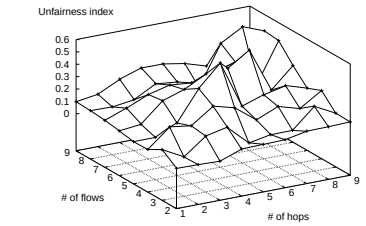
Evgeny Osipov and Christian Tschudin
University of Basel
Computer Science Department
Bernoullistrasse 16, CH–4056 Basel, Switzerland
Email: {evgeny.osipov | christian.tschudin}@unibas.ch

Abstract

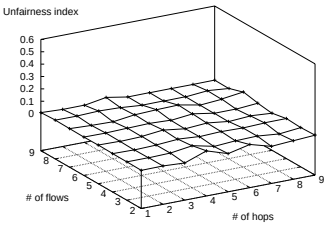
Knowing or being able to measure the "path density" at sources of communications is essential to provide fair capacity distribution between sessions in multi-hop ad hoc networks. We propose and have implemented PDP, a path density recording protocol. In this paper we describe PDP, discuss protocol design options and explain how it was piggybacked onto an existing reactive routing scheme. We assess the validity of our approach both using simulations and real world measurements.

1 Introduction

In [1] we presented an adaptive distributed capacity allocation scheme for multi-hop wireless networks. We showed that by throttling the output rate at ingress nodes we achieve both an increase in total network throughput and almost perfect fairness. Figure 1, generated from simulations, shows the degree of improvement with respect to unfairness between multiple TCP sessions in static networks with different maximum number of hops. When our rate bound is implemented we observe that the unfairness virtually vanishes.



A. Plain TCP over IEEE 802.11



B. TCP over IEEE 802.11 with our ingress throttling scheme (see Appendix A)

Figure 1. TCP unfairness index (simulations). See Appendix B for the definition of "unfairness index" and see Figure 5 for the topologies.

In our solution [1] the throttling level is a function of: (a) the number of hops for a particular "connection" (routing path between two peers) and (b) the number of competing connections on the path (*path density*). In the case that a reactive ad-hoc routing protocol is used to provide connectivity, the first parameter of interest is easily obtainable from the route reply (RREP) message received by the source. In this paper we show how the path density parameter can be obtained at run-time during the route establishment phase.

The concept of a path density (PD) implicitly exists in the major quality of service (QoS) architectures developed for the Internet. For instance, in services which require a before hand reservation of network resources, this information can be extracted from the state (either aggregated or per-flow) established by a resource reservation protocol like RSVP. In the simplest case we can count the number of entries identifying the ongoing flows in every router on the path of a particular session and report this number to the source. In MANETs, however, obtaining the path density information is difficult due to frequent topology changes and the specifics of the transmission medium. In MANETs, the competing connections might not share common nodes but still do compete with other connections in the carrier sensing range.

1.1 Contribution and structure of the paper

We developed two variants of a distributed path density protocol: A *stateless* PDP and a *state-full* PDP. To the best of our knowledge, PDP is the only scheme which allows the on-demand gathering of an ad-hoc network state and which was assessed in simulations as well as real world experiments. The design of PDP greatly benefited from being able to instantly switch between simulations and real world experiments during the developing, debugging and performance measuring phases.

The rest of the paper is organized as follows. In Section 2 we introduce the problem of path density gathering and describe major design options for PDP. In Section 2.5 we show how PDP was piggybacked onto an existing routing protocol. After that in Section 3 we report on performance results of PDP obtained using simulations and real-world measurements. We discuss open issues and related work in Section 4 before concluding with Section 5.

2 Measuring the path density

Throughout the paper when talking about a stream of packets between applications at a source and a destination node we will use the terms *connection*, *session* or *flow* interchangeably.

2.1 Problem statement

The problem addressed in this paper is formulated as to discover the number of connections competing for the transmission medium along a path of a particular session. The problem is illustrated in Figure 2. In the figure, Connection 1 is our sample connection which attempts to discover the *path density* along its path. The correct scheme should report five cross-connections plus the main Connection 1 itself. When any two flows partly share their forwarding paths (as is the case with flows one and four in Figure 2) the common nodes should treat them as different connections. However when two or more connections completely share the path from a source to a destination the forwarding nodes should treat this case as one end-to-end session. In the later case the source nodes should perform additional shaping actions as described in [1]. The forwarding nodes of Connection 1 should also take care of distributing the known information about ongoing connections in the corresponding one-hop region. In our example this is needed to introduce the presence of flows two and three to five and six and vice versa, since their forwarding nodes in general might be outside the communication regions of each other.

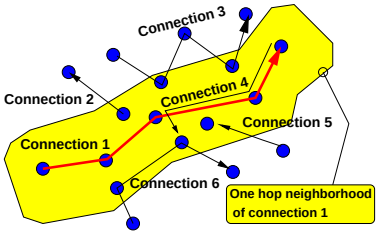


Figure 2. Counting competing connections.

2.2 General solution scheme

We developed two different approaches for gathering the path density information: A *stateless* route request (RREQ) driven and a *state-full* route reply (RREP) driven scheme. Further on we will refer to the first scheme as SL-PDP (StateLess PDP) and to the second scheme as SF-PDP (StateFull PDP). In both approaches every node in the network maintains one state variable *ND* (the neighborhood density). This is the number of cross-connections in the neighborhood of two hops. With every new connection appearing in the neighborhood this value is incremented by one. The state is periodically aged and is decremented by the number of connections which were silent during this period.

The stateless approach does not keep a per-connection state in the forwarding nodes and utilizes the fact that a *route request* message indicates the desire of a node to communicate with another node. In this scheme every RREQ issued by the source advertises the presence of the connection to all nodes which receive its original or rebroadcasted copy. There are two major weaknesses of this scheme that we discovered when experimentally assessing the functional correctness. Firstly, SL-PDP also counts the failed connection setups as existing connections and maintains this information at least for the duration of the aging timer. Secondly, since flooding is used for dissemination of RREQs we cannot control the range of their distribution. As a result, the information about the presence of a connection is spread over more than two hops. We do not discuss further the functionality of SL-PDP in this paper and refer to [2] for more information.

The statefull approach is free from these weaknesses since it counts only active connections and controls the range of information dissemination. The main idea of this scheme is that on reception of a *route reply* message all involved nodes establish a state corresponding to the end-to-end session. A node delays the announcement of a connection to the neighborhood until it sees the first data packet from this connection, as only this event indicates that the connection is active. The announcement of active connec-

tions in one hop neighborhood is done every time a node sends or re-broadcasts a route request message. We now describe SF-PDP in more details.

2.3 Statefull path density gathering

In SF-PDP we identify presence of a communication session between two nodes in the network by a source-destination pair (SD). In this scheme every node maintains a table of known SD pairs (*SD_table*). The entries in this table can be of four types: (a) Local active, (b) local inactive, (c) one-hop active and (d) two-hops active as explained in the following subsections. The ND state variable is the number of all "local active" and "non-local active" entries in the SD_table. The major operation of the statefull PDP is shown in Figures 3 and 4.

2.3.1 Adding "local inactive" SD entry

The operation of the statefull PDP during registration of a connection in the network is shown in Figure 3. When a route request message for a particular connection successfully reaches the destination the RREP message is returned to the source. However, reception of the RREP message by a node does not indicate the success of the route establishment procedure since the message itself can be lost on the path to the source. SF-PDP treats the event of RREP reception as an indication of a possible connection through this node. Upon reception of a RREP a SD entry corresponding to this connection is inserted to the SD_table and is marked as inactive.

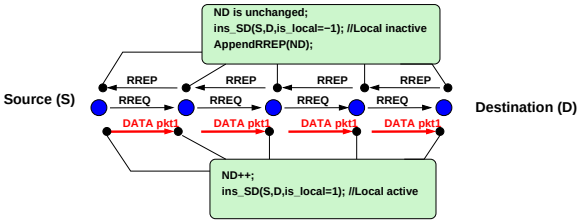


Figure 3. SF-PDP: Processing of RREP and a first packet.

2.3.2 Adding "local active" SD entry

The only definite indication of the active connection in any forwarding node is the event of reception of the first data packet. Therefore, when a local inactive SD record is created, the forwarding engine is instructed to catch the first packet in either direction of the connection.The SD entry becomes active only when the first data packet arrives to the node.

2.3.3 Adding "non-local active" SD entry

The non-local active entries are those which identify connections ongoing in the neighborhood of two hops but not passing through this node. The information about such connections is distributed with broadcast route request messages for any connection originated or re-broadcasted from any of the one-hop neighbors. Figure 4 illustrates the principle of such information dissemination. Before issuing or re-broadcasting the route request message, SF-PDP examines the local SD_table and appends all "local active" and "one-hop active" SD pairs to the RREQ_SD_List. After that it piggybacks the constructed list to the RREQ messages and transmits it.

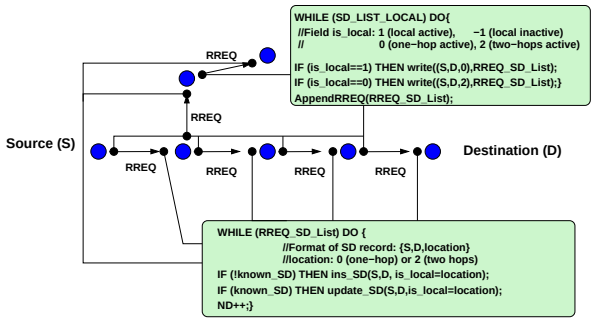


Figure 4. SF-PDP: Processing of RREQs.

When a route request message is received by a node, the SD entries from the RREQ_SD_List are inserted to the local SD_table and marked as "non-local active" of the corresponding location. After that SF-PDP removes the RREQ_SD_List from the received RREQ message and appends its own list as described above. By replacing the RREQ_SD_List with every retransmission we do not allow the information about local connections to spread further than one hop.

2.3.4 Aging of SD entries

Every entry in SD_Table is assigned a timer. The entry is deleted when the timer expires. The duration of the timer is the same as of the route expiration timer.

2.4 End-to-end density reporting and smoothing

The overall goal of PDP is to deliver the path density information to the source nodes. This information is used by the interface queue to configure the delay of emission of locally generated data packets as described in [1].

The path density is the maximum value of ND variables from all forwarding nodes on the path of a connection. The ND value is piggybacked in the route reply message at the destination node and can be modified by every node which receives and forwards the RREP message. When a forwarding node receives a route reply message, PDP examines the local ND state and compares its value to the ND field in the RREP message. The higher value proceeds further to the source.

As we see the path density state is refreshed with every route reply message. Therefore the frequency of its update depends on the frequency of route refreshes initiated by the source. Here we assume that route refreshes are periodically initiated. In the next subsection we discuss how this requirement can be combined with reactive ad-hoc routing protocols.

Due to high dynamics with which new flows might enter and leave the ad-hoc network, the path density reported to the source may fluctuate in time. Therefore, in order to avoid reconfiguring the scheduler on the interface queue too often we allow sources to *smooth* the path density by computing a sliding average.

2.5 Integration in ad-hoc routing protocols

Our primary goal for the implementation of PDP was to avoid introducing additional message exchanges to gather the path density information. This is because from the functional point of view the operations of PDP are very similar to operations of the route establishment phase. Implementing an additional stand-alone protocol for dissemination of PDP information would in the worst case double the capacity already consumed by a routing protocol. We decided to re-use the existing mechanism of message dissemination of the ad-hoc routing schemes.

We integrated PDP in LUNAR [6], which is a L2.5 reactive routing scheme. We also considered possibility of PDP integration in other reactive routing protocols, i.e AODV [4] and DSR [5], and in the proactive OLSR protocol [3]. In this subsection we present our observations on the possibility to use AODV, DSR and OLSR protocols as a PDP carrier.

The route refresh period in proactive routing is normally larger than in reactive schemes. In OLSR, for example, the default refresh period is 16 seconds. We consider it too large assuming high dynamics with which new sessions might enter the ad-hoc network. Based on these observation we do not see proactive routing protocols suitable for dissemination of PDP information.

As for AODV and DSR, we see a number of difficulties for possible integration of PDP in these protocols. As stated in Section 2.1, the flows which share a part of their path to the destination should be treated by our scheme as separate connections. This requirement places a major limitation on the carrier protocol: It should not perform path aggregation. However, this functionality is embedded in both AODV and DSR. Firstly, path aggregation is the normal operation for routes to the same IP subnet. Secondly, even for the cases where IP level aggregation is not used, AODV and DSR provide gratuitous route replies: When a forwarding node is part of the path to a particular destination, it may return the RREP to the source without further propagation of the route request message. Integration of PDP into AODV or DSR would require changes to the core functionality of these L3 routing protocols.

2.6 Integration of PDP in LUNAR

LUNAR is a L2.5 reactive routing scheme. The major difference of LUNAR from AODV and DSR is its position in the TCP/IP protocol stack. Since LUNAR is located below IP layer, the IP level route aggregation which is present in L3 routing schemes is impossible. The route request messages in LUNAR always propagate from the source to the destination. That is, gratuitous replies are not used either. LUNAR is well suited for integration with PDP also because of its high dynamics of information update: LUNAR re-establishes the entire path every three seconds.

3 Experiments

The single code base both for the Linux kernel and the network simulator ns-2 [12] allowed us to debug and extensively test the PDP+LUNAR functionality on a wide range of static and dynamic simulation scenarios. We implemented and tested both stateless and statefull PDP schemes. Once we obtained a stable protocol we generated a real world version and performed a correctness test in a test-bed as described in Appendix C. While performing a preliminary evaluation of SL-PDP in the simulator we found the drawbacks of this scheme described in Section 2.2. As a result we decided to go on with SF-PDP. In this section we present the results from testing the protocol both in simulations and the real-world test-bed.

3.1 Metrics

In all experiments we used correctness and convergence time metrics to evaluate the performance of PDP. Correctness is evaluated with respect to the “path density” as reported to the end nodes of the session (the path density in our experiments was also reported to destinations because we used bidirectional traffic in the experiments).

3.2 The effect of path density (Simulation)

First, we show the effect of path density on the unfairness index and total TCP throughput in networks with different number of active TCP flows. We performed a series of simulations in ns-2 (see Appendix C for setup details) on a set of three hop networks depicted in Figure 5. We varied the number of connections in the network from 2 to 9; For each case we run 30 simulations with enabled and disabled reaction on the reported path density. The dynamics of the unfairness index (see (2) in Appendix B) in both cases is shown in the part of surfaces corresponding to three hops and 2 – 9 connections in Figure 1.

Under our scheme the fairness among competing TCP flows was close to perfect, while ignoring the path density we observed an unfairness of up to 25%. Note that the nature of the used unfairness index reflects the closeness of individual goodputs of each flow to the ideal share, therefore the closer the unfairness index to zero the higher is the minimal individual goodput.

Number of connections	Total TCP throughput, kb/s	
	Without path density	With path density
2	419	419
3	405	407
4	396	407
5	395	406
6	394	406
7	386	407
8	375	406
9	380	406

Table 1. The effect of path density on total TCP throughput.

Table 1 shows the effect of ingress rate throttling on total TCP throughput depending on the path density for the two cases. As we observe from the table the total TCP throughput in the network is higher when accounting for the path density. This implies better network utilization when all competing flows fully utilize their allocated share. Note that in reality we will also have a number of short-living communications and flows with a number of short transmission bursts e.g. web browsing. In this case the share of capacity computed based on the path density reported by PDP will not be fully utilized. When several connections are originated from the same source the leftover capacity can be reused by the active flows, otherwise the network will be underutilized until the route for the inactive connection will be removed. Therefore it is essential to dynamically update the ingress nodes with the path density information. In the case of our PDP+LUNAR combination the information about inactive flows will be aged within three seconds in all PDP-aware forwarding nodes.

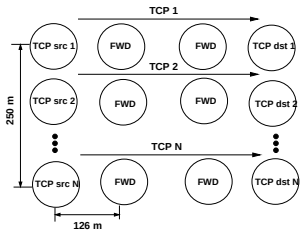


Figure 5. A set of three hop networks for illustration of the path density effect.

3.3 Static Scenarios (Real World)

We present an evaluation of the functional correctness of PDP on two static scenarios depicted in Figure 6. While all nodes in reality are located in the reception range of each other (hence the same radio interference domain) we configured LUNAR so that a node can hear the data transmission only from a selected set of nodes and discard packets from others. In this sense only Nodes 3 and 4 in Figure 6*a* "share" the regions of assured data reception of each other. In the second case (Figure 6*b*) Node 2 is able to hear transmissions from Node 4 and Node 3 from Node 5. PDP should, however, report the same density information on all nodes.

In both scenarios Session 1 started at time 1 s and finishes at time 30 seconds. Session 2 establishes the path 10 seconds after Session 1. The duration of Session 2 is 10 seconds. We used the ping protocol to generate traffic of the corresponding sessions. The interval between two consequent ICMP requests is 100 milliseconds. We repeated both real world experiments up to 10 times and obtained the same behavior of our two metrics. Figure 7 shows the results from one of these runs.

As is visible from the time diagrams, PDP correctly reports the path density to the corresponding end nodes of the two connections within 1.5 – 3 seconds from the start time of a session. The delay in dissemination of the information is explained by the default route refresh interval in LUNAR. After the session is established the earliest time a new RREQ message is generated by the destination in the backwards direction is after 1.5 seconds.

3.4 Dynamic Scenario (Simulation)

We evaluated the behavior of PDP in presence of mobility in simulations only. This time we seek a quantitative assessment of the path density information provided by PDP. We studied the scenario shown in Figure 8.

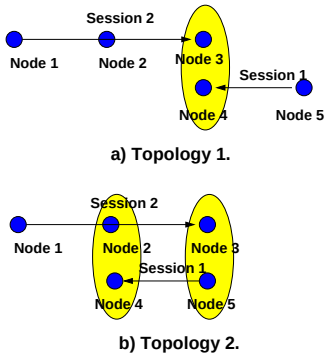


Figure 6. Topologies for real world-experiments.

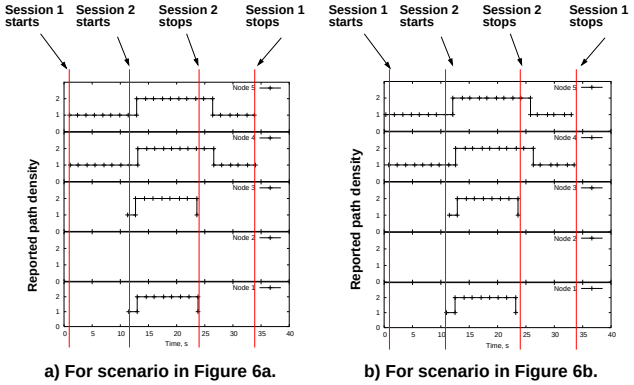


Figure 7. Reported path density in real-world experiments.

In the scenario we have three TCP sessions each following a path of two hops. Initially all nodes are located outside the range of assured reception of each other. After five seconds from the simulation start the middle nodes of sessions one and three begin a movement towards the middle node of Session 2. The speed of the nodes is 10 m/s. In their final destination both Nodes 2 and 8 are in the range of assured reception of each other and Node 5. The mobile nodes remain in this position for 80 seconds; After that they start moving back to their original position.

The goal of this experiment is to evaluate the dynamics of PDP based source throttling. We performed two experiments on this scenario. First, the path density information was ignored and TCP flows might transmit without rate limitation. In the second experiment we configured the scheduler on the interface queue with a throttling bound dynamically computed according to the path density information reported by PDP. Figure 9 presents the results of these experiments.

As we observe from the figure, the slope of TCP sequence numbers curves is the same for all three flows provided path density is taken into account, which results in fair sharing of the network's capacity. Moreover, in this case the progress of all TCP flows is smooth and free from interruptions when compared to the case where the path density information is ignored. In addition to the smoothness of the TCP flows, the resulting total TCP throughput (not shown in the figure) remains the same as in the case without rate limitation at sources.

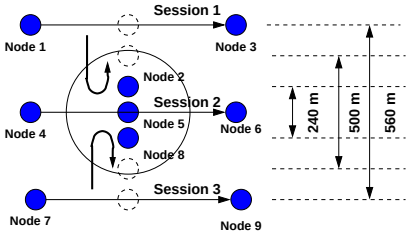


Figure 8. Topology 3 for experiments on a dynamic scenario.

To sum up, the results of the last experiments illustrate our overall goal: If flows in a MANET are aware about the presence of each other and reduce their rate accordingly, none of the competitors is able to capture the capacity as it happens for the plain combination of MAC 802.11 and TCP. The latter case is represented in Figure 9 by flow TCP 2 (w/o path density) which worsens the performance of TCP flows 3 and especially 1.

4 Discussion and related work

Our path density gathering scheme presented in this paper is a distributed protocol for the on-demand discovery of an ad-hoc network state. The PDP protocol plus our ingress rate throttling approach permits to implement a capacity allocation scheme with guarantees on fairness of communications.

In addition to the admission control in sources we see potentials of our protocol for congestion-aware routing. Indeed, the ND state kept in all forwarding nodes is an excellent indicator of the neighborhood's load. If a forwarding node – instead of re-broadcasting the newly arriving route request – would first examine the neighborhood density, it can estimate the chance for this flow to obtain an acceptable service while being forwarded through this area. If a neighborhood is overloaded with existing connections the route request can simply be discarded. Assuming a network with relatively high node density, the "surviving" route requests would discover less loaded paths.

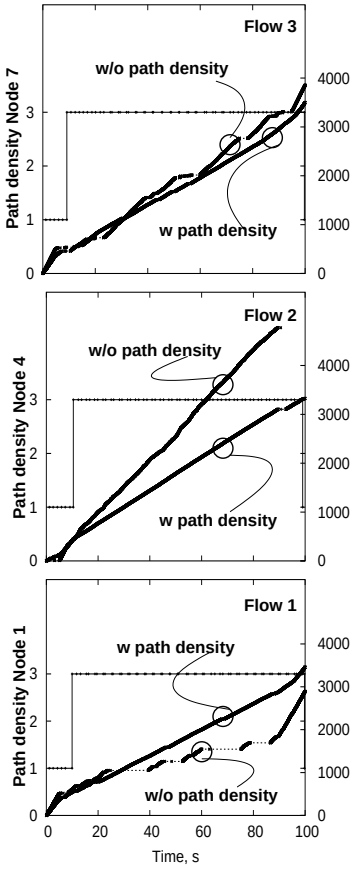


Figure 9. Path density and TCP sequence numbers progress for scenario in Figure 8.

4.1 Related work

The idea of estimating the state of an ad-hoc network is not new. There are several approaches based on distributed network state exchange. In [7] SWAN, a distributed QoS architecture for MANETs is proposed. The proposed admission control at sources as well as the rate limitation scheme in all nodes is based on hop-by-hop measurements of the available bandwidth. The authors suggest to use a reactive routing protocol to gather the results of these measurements along a path of a connection. New flows are admitted for transmission only if the reported path capacity is less than a certain threshold.

In [8] the authors suggest to exchange the occupancy level of the interface queue of every node in one and two hops neighborhood. This information is then used to construct a virtual queue of the neighborhood and to coordinate dropping of packets. The exact mechanism to exchange such information is however not described in the paper. The authors suggest to use additional message exchanges for the dissemination of the information.

In [9] the authors discuss a distributed weighted fair queuing algorithm, similar to its analog in the fixed Internet. In order to distribute the weights to the interface queue of each node in the neighborhood of one hop, the authors suggest to encode this information in the MAC 802.11 header.

5 Conclusions

In this paper we considered the problem of measuring the "path density" in MANETs. We presented an approach for gathering such information in a distributed way at run-time. We understood that building PDP based on observing route request transmissions only leads to incorrect counting of the failed connections and resorted to storing per-connection state in the forwarding nodes. We showed how PDP can be piggybacked inside LUNAR messages without introducing new transmission events and tested our implementation with combined simulation and real-world measurements.

Acknowledgments

We wish to thank an anonymous reviewer for the detailed comments which greatly helped the preparation of the camera-ready version of this paper.

References

[1] E. Osipov, C. Jelger, Ch.Tschudin, "TCP capture avoidance in wireless networks based on path length and path density", Technical Report CS-2005-003, University of Basel, Switzerland, April 2005.

[2] E. Osipov and Ch. Tschudin "A path density protocol for MANETs", Technical Report CS-2005-004, University of Basel, Switzerland, May 2005.

[3] T. Clausen, P. Jacquet, "Optimized link state routing protocol (OLSR)", RFC 3626, IETF, Oct 2003.

[4] C. Perkins, E. Belding-Royer and S. Das, "Ad hoc on-demand distance vector (AODV) routing", RFC 3561, IETF, Jul 2003.

[5] D.B.Johnson, D.A.Maltz,Y-C.Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)", IETF draft (work in progress), 2003. [Online]. Available: `http://www-2.cs.cmu.edu/~dmaltz/dsr.html`.

[6] Ch. Tschudin, R. Gold, O. Rensfelt and O. Wibling, "LUNAR: a lightweight underlay network adhoc routing protocol and implementation", In *Proc. NEW2AN'04*, St. Petersburg, Russia, Feb. 2004.

[7] G-S. Ahn, A. T. Campbell, A. Veres, L. Sun, "SWAN", IETF draft (work in progress), 2003. [Online]. Available: `http://comet.ctr.columbia.edu/swan/draft-ahn-swan-manet-00.txt`.

[8] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED" , In *Proc. MobiHoc'03*, Annapolis, MD, USA, 2003.

[9] N.H. Vaidya, P. Bahl and S. Gupta, "Distributed fair scheduling in a wireless LAN", In *Proc. MobiCom'00*,Boston, MA, USA 1999.

[10] K. Xu, S. Bae, S. Lee and M. Gerla, "TCP behavior across multi-hop wireless networks and the wired Internet" , In *Proc. WoWMoM'02*, Atlanta, GA, USA, Sep. 2002.

[11] Uppsala University Ad Hoc Implementation Portal, [Online]. Available: `http://core.it.uu.se/AdHoc/ImplementationPortal`.

[12] Network Simulator NS-2, [Online]. Available: `http://www.isi.edu/nsnam/ns/`.

Appendix A - Adaptive capacity allocation scheme for capture-free communications

TCP capture, as is described in [10], is one of the unsolved problems in multi-hop ad hoc networks which results in extremely unfair distribution of network bandwidth between competing sessions.

In [1] we suggested a scheme for dynamic capacity distribution between competing connections that partly or completely share communication regions. We proposed the ingress throttling mechanism which guarantees capture-free communications for all involved flows.

The input information to our scheme is the *number of competing connections* on the path of a particular multi-hop TCP connection (*path density*) and the knowledge of the saturation point of the radio transmission medium beyond which a single TCP flow starts to loose packets (*critical network load*). This information is used to compute the output rate at ingress nodes (1).

$$r_{ingress} = \frac{\alpha \cdot L_{TCP}^{crit}}{h \cdot N}. \quad (1)$$

In (1) L_{TCP}^{crit} is the estimated rate of TCP data segments which creates the critical network load, h is the number of hops of a particular flow and N is the path density. If all sources follow this rule, then the total network capacity is fairly distributed between the competing flows as shown in Figure 1.

Appendix B - TCP unfairness index used in Figure 1

We define the unfairness index u used in Figure 1 as the normalized distance (2) of the actual throughput of each flow from the corresponding optimal value.

$$u = \frac{\sqrt{\sum_{i=1..n}(X_{opt_i} - X_{act_i})^2}}{\sqrt{\sum X_{opt_i}^2}}. \quad (2)$$

In this formula X_{opt_i} is the ideal throughput of flow i obtained under fair share of the network capacity. In order to compute this value we divide the throughput of the corresponding flow obtained when running alone in the network by the number of competing flows. X_{act_i} is the actual throughput of the same flow achieved while competing with other flows. This index takes the values between 0 and 1 and reflects the degree of global user dissatisfaction, hence the value of 0 corresponds to perfectly fair communications and 1 represents the opposite case. Note that the index is useful only when applied to two or more flows. This is also reflected in Figure 1, where the y-axis starts from two connections.

Appendix C - Experiment Setups

The real-world results were obtained from a setup of five DELL Latitude laptops with ZyXEL ZyAIR B-100 wireless interfaces. We use the Linux operating system with 2.6 kernel and LUNAR implementation available from [11].

The simulation results were obtained using *ns* simulator of version 2.27. In simulations with scenarios depicted in Figures 5 and 8 we used TCP Newreno as the most popular variant of the protocol. We set the value of the TCP maximum segment size (MSS) to 600 B. The data transmission rate of all devices is 2 Mb/s. Other ns-2.27 parameters have default values.

Interactions between TCP, UDP and Routing Protocols in Wireless Multi-hop Ad hoc Networks

Christian Rohner, Erik Nordström, Per Gunningberg, Christian Tschudin[b]

Uppsala University, Sweden and University of Basel, Switzerland[b]
{chrohner|erikn|perg}@it.uu.se, christian.tschudin@unibas.ch

Abstract

The achieved throughput in an ad hoc network is affected by many factors, including radio interference between hops, the ability of the routing protocol to react on topology changes and a complex interaction between the application and underlying protocols. This paper studies experimentally the impact of these factors on UDP and TCP throughput. Furthermore, when both TCP and UDP share some hops in an ad hoc network there is a complex interaction between the transport protocols as well as with the routing protocol. Our results show that this interaction results in significant UDP jitter, instable routes and significantly lower TCP throughput. We use a controlled real testbed for the experiments and a graphical tool that captures the interactions.

1 Introduction

It is well known that TCP suffers in a wireless environment because it incorrectly interprets packet loss as congestion. It is also well known that both UDP and TCP performance is reduced in a multi-hop 802.11 environment because of radio interference between hops. Simulation results by Holland et al. [5], show that the radio interference for two hops reduces the possible throughput to 50 percentage. For three hops the throughput is 33 percentage. This result was supported both by simulations and a theoretical model.

Another performance impairing factor is frequent routing updates. In ad hoc networks with mobile nodes, the problem for TCP is getting more severe since it is likely that packets get lost when a route is lost and the routing protocol needs to discover a new route.

Finally, it is also known that UDP is unfair to TCP when they share a common bottleneck link. TCP will back off and try to find a sending rate that adjusts to the rate of UDP. This adjusting time can be long depending on end-to-end delay and assumes some stability in available bandwidth. All these factors impact the performance of an ad hoc network and have been individually studied, but the interactions between them in a dynamic environment is not generally understood.

The intention with this paper is to demonstrate measurements results and to study unforeseen interactions between components at different layers that negatively impact the experienced application level performance in the ad hoc network. With our tools we can capture the series of transmissions involved and explain the timing of events relative to the node movements. We do this in a controlled experimental environment with the possibility to repeat experiments to understand the variance. The scenarios are carefully chosen to reflect the intended interaction between protocols, radio range, movement and prospective applications.

All scenarios are derivatives of the scenario shown in Figure 1. It comprises three nodes that have contact only with their adjacent neighbors. A low rate UDP flow, traversing two hops, is sent from node 2 to node 0. This scenario constitutes our baseline case, which remains static and is never changed in any of the scenarios. We then add a fourth node, sending an additional TCP flow to node 0. We also add dynamic routing and mobility. We do this to study the effect on the baseline case in terms of interference, contention for the wireless channel and buffer space. The low rate UDP flow is used to sample the wireless channel so that we can infer the interference from the fourth node. In theory, the expected result would be that the fourth node will have a limited

effect on the UDP flow, because TCP should adapt to the remaining bandwidth.

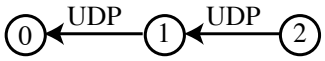


Figure 1. Base scenario: A static two-hop setting with a constant bit rate UDP flow.

From our measurement results we find that adding the fourth node interferes with UDP, inducing jitter and packet loss. Adding mobility and dynamic routing causes an average packet loss of up to 21.6% on the UDP flow. The results indicate that TCP and UDP contend for both buffers at nodes and radio transmission time. Both protocols suffer from significant loss as expected, and TCP backs off. Somewhat unexpected is that UDP losses are higher than could be expected on shared wireless links. This raises the question if TCP can adapt quickly enough to the environment where there is node movement and connectivity changes.

Another observation is that the routing protocol is also affected by the TCP flow, preventing the establishment of correct routes during extended periods. As TCP will back off and then start to probe again for the sustainable bandwidth, this potentially leads to long term oscillations overlaying the short term interactions among the protocols.

The paper is structured as follow. In section two we discuss related work. In section 3 we describe our experimental set-up, scenarios, experiments and results. We conclude with a section on discussion and further work.

2 Related Work

The TCP feedback loop is responsible for adapting the sender data rate in response to, e.g., congestion. However, this end-to-end congestion control mechanism has reduced efficiency in wireless networks because transmission is inherently broadcast. Furthermore, there are different and transmission rate dependent ranges for unicast transmission, broadcasts and interference [2].

Holland et al., examine in simulation the TCP performance over mobile ad hoc networks. They note that TCP suffers significantly in mobile, multi-hop scenarios, simply because TCP can not distinguish between link failures and congestion.

In [3], Gerla et al. study through simulation, the interactions between TCP and different MAC layers. Our work complements that work by also looking at other layers and different transport protocols.

Gupta et al. report in [4] on decreased TCP performance in the presence of interacting UDP flows. Their study is done in the ns-2 simulator on an artificial grid topology and with relatively high UDP data rate (800 Kb/s). Since UDP lacks rate adaptation and congestion control, a high data rate CBR flow will naturally congest the channel. Our work complement that work by performing real world measurements with UDP and TCP data flows in scenarios with increasing complexity. Moderate data rate UDP is used (typically streaming music) which – as we believe – is more realistic

Internet routers improve fair access to its resources by using Random Early Detection (RED). Xu, Gerla, Qi and Shu study in [8], TCP fairness in wireless ad hoc networks. They look at queuing policies to improve TCP fairness and show that the RED scheme fails, because flows compete not only for queue space but also for the wireless channel. They propose Neighborhood RED (NRED) where all individual queues of nodes within a neighborhood is treated as a shared distributed queue. Based on the notion of this distributed queue, a forwarding node can drop packets to increase TCP fairness.

3 Experiments

This section reports on real world experiments that examine the contention between TCP and UDP data flows. After providing some setup details, we discuss a sequence of increasingly more complex scenarios and the results of the following experiments.

All experiments were conducted indoors in a systematic way using the APE testbed [7]. APE provides a test environment that allows repeatability of experiments and provides support for extensive logging and analysis of experiment data. The APE testbed orchestrates the scenarios and provides verbose logging on the network interface and network layer. Standard laptops (IBM X31) with Lucent/ORiNOCO IEEE 802.11b network interfaces are used, transmitting at a fixed rate setting of 11 Mbit/s, without RTS/CTS. The interface used a standard MTU setting of 1500 bytes. Iperf and Ping were used to generate data. For UDP, 1470 bytes data

was sent[1], resulting in 1512 byte frames in the ether. This size was used to compare against the normal TCP segment size. Logged and time stamped data is time synchronized between nodes using periodic broadcasts of time information at a rate of one packet every 10 seconds. This time information is used to synchronize the experiment data collected from the different nodes.

The AODV-UU [1] implementation was used where dynamic routing was needed. AODV-UU was chosen because it is mature and has been interoperability tested.

3.1 Bandwidth Measurements

As a calibration and verification to prior work, we start by measuring the maximum achievable throughput for TCP and UDP data flows. We do this to establish the expected bandwidth for our baseline scenario, in single and multi-hop settings without mobility. For this experiment, four nodes are placed in a linear chain such that a node can only communicate with its adjacent neighbors. The UDP and TCP throughput over one to three hops are measured. The results are shown in Table 1. We differentiate in the one hop measurements between the two nodes located next to each other (short hop) and located in the periphery of each others transmission range (1 hop).

[Mbit/s]	short hop (10 cm)	1 hop (ca. 20 m)	2 hop	3 hop
UDP	5.64	4.85	3.60	2.01
TCP	3.71	1.68	0.78	0.62

Table 1. The maximum throughput is decreasing with increasing number of hops.

The UDP throughput is as expected higher than the TCP throughput, since UDP is one way traffic without acknowledgments that compete for transmission time over the shared channel. The maximum achievable TCP throughput over one hop is roughly a third of that of UDP. Acknowledgments going in the reverse direction is not as large as the data packets in the forward direction, giving more channel time for the data.

When increasing the number of traversed hops, the throughput decreases for each hop, in case of TCP following roughly an $1/n$ slope where n is the number of hops. This result matches the simulation results from Holland, et al. [5].

A noteworthy observation is the significant difference in one hop throughput between nodes that are placed next to each other (short hop, 10cm), and nodes that are located in the periphery of each others transmission range. In our indoor environment, that was a separation of 20 m. An analysis of the TCP experiments shows a slightly higher variation in the round trip time for separated nodes, indicating retransmissions on the link-layer as one reason for that artifact. TCP is more affected than UDP because of its own adaptation mechanisms that are not designed for fluctuating wireless environments.

3.2 Baseline Scenario: Static Multi-hop UDP

This experiment presents our baseline scenario shown in Figure 1. It consists of three nodes 0, 1, and 2 placed in line. Each node has connectivity only with its adjacent neighbors. End node 2 sends a constant bit rate UDP flow to node 0, over the intermediate node 1. As well as establishing the baseline performance, the experiment intends to examine the interference between the consecutive hops in the multi-hop path.

For each test run, the *offered* data rate (rate of data from the application) is increased, permitting to study the effect of interfering hops on the multi-hop path. Since the bandwidth at node 1's network interface is shared between the two "links" we expect the overall UDP throughput to decrease when the offered data rate at node 2 exceeds one half of the bandwidth at node 1.

Figure 2 shows the *received* data rate as a function of the *transmitted* data rate (rate of the data actually sent on the channel). It can be observed that the transmitted data rate for this two hop path achieves the best received data rate at around 3.6 Mbit/s. Increasing the offered data rate beyond that has a negative impact on the overall throughput, because of interference between the hops. Note that although we increased the offered data rate up to 10 Mbit/s, the *transmitted* rate never exceeded 5.2 Mbit/s.

After having determined the interference limited data rate over two hops, the UDP data rate was fixed at 192 kbit/s. This is well within the limit of link interference and also resembles the rate of an MP3 data flow. We expect such a low rate flow to have close to 100% delivery ratio in our baseline scenario with no interference.

The average UDP delivery ratio over five test runs was 99.2 %. It could be verified that packet losses oc-

[1] 1470 bytes is the default Iperf setting.

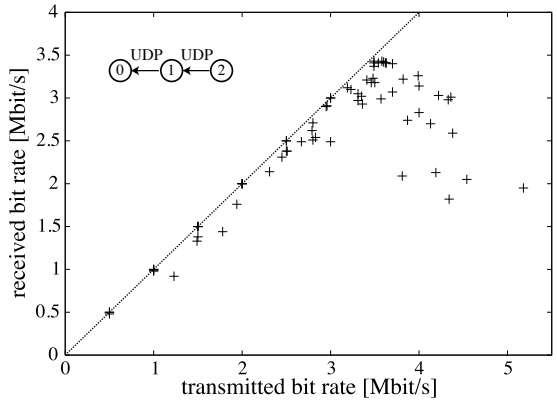


Figure 2. Received bit rate as a function of the transmitted bit rate in the two-hop scenario for offered bit rates between 0.5 Mbit/s and 10 Mbit/s (with 0.5 Mbit/s intervals in between). The line shows the optimal throughput when there is no packet loss.

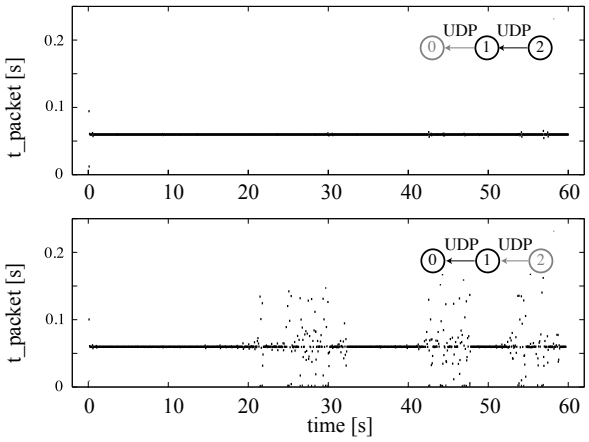


Figure 3. Example of UDP packet interspacing time in the static multi-hop scenario (192 kbit/s).

curred at the second hop link between node 1 and 0.

The interaction between the two hops is analyzed by looking at the UDP packet interspacing time. A variation in the packet interspacing time indicates either contention for medium access, link layer retransmissions, or packet loss. Figure 3 shows the packet interspacing time for the UDP flow between node 2 and 0 for a representative run. We observe that the second hop is more affected by interactions than the first link. However, as expected, the packet loss is low and packet interspacing small. In the following experiments we use these baseline results to study the effects on the UDP flow by adding an extra node, a TCP flow, dynamic routing and mobility.

3.3 Multi-hop UDP With an Interfering TCP Flow

In this scenario we extend the baseline scenario with an interfering TCP flow. An extra node 3 starts at the far left position in Figure 4, outside the transmission range of node 1. Ten seconds into the scenario it starts a TCP connection to node 0. After another ten seconds, node 3 starts moving toward node 0 where it stops its movements at time 38 but continues to send data. At this position, node 3 is within the transmission range of node 1 and hence will interfere.

Routing in this scenario is static. The purpose is to see how TCP adapts its send rate (if at all) when moving to a position where it is potentially more affected by channel contention from both node 1 and node 2, without an actual change in the path. We also want to see how the UDP flow from node 2 to node 0 is affected by the increased contention. We do this to better understand the effect of mobility and spatial placement of nodes.

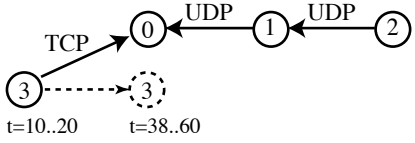


Figure 4. Scenario for two-hop UDP with an interfering one-hop TCP flow.

Over five test runs, the average UDP delivery ratio was 97.4 %. Although a slight decrease from the previous scenario, it does indicate that the UDP flow is not significantly affected by the interference from the TCP flow.

In Figure 5, we see the UDP packet interspacing time on the link between node 2 and 1, and 1 and 0 respectively, along with the TCP time sequence graph for the TCP flow between node 3 and 0, for one of the experiments. Although variations are apparent throughout all experiments, Figure 5 is representative for a typical test run.

As expected, the variation of the packet interspacing

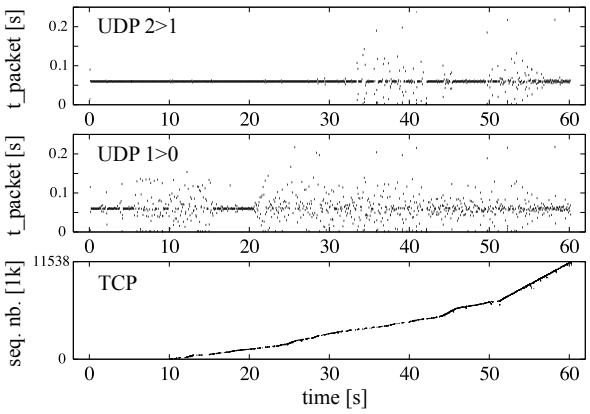


Figure 5. Example of UDP packet interspacing time and TCP sequence graph.

time between node 2 and 1 is more apparent after node 3 is in direct contact with node 1. It interesting though, that the packet interspacing time between node 1 and 0 tends to stabilize during the static periods (time 5..10, and after time 50 where basically the variation pattern from the first hop gets propagated).

The TCP flow itself is also affected. In the time sequence graph we observe some interruptions in the TCP flow of up to 500 ms, in particular during the movement of node 3 (time 20..38). Furthermore, TCP throughput is unstable even during the stationary phases at the beginning and end of the experiment. While the maximum achieved TCP throughput (1.63 respectively 3.37 Mbit/s) matches the one-hop figures of Table 1 where no UDP background traffic was present, the average throughput during these phases is significantly lower (0.89 respectively 1.66 Mbit/s).

An interesting observation can be made at time 50, when there is an immediate increase in the TCP throughput. This behavior is persistent in most of the experiments. It is not clear why TCP increases its throughput in this situation, at the same time as the fluctuations on the second UDP hop seem to stabilize. One possible explanation could be the capture effect [9]. Understanding the exact reasons for this particular behavior requires further investigation.

3.4 Multi-hop UDP Sharing Hops with a TCP Flow

This scenario, called “Roaming node”, extends the complexity of the baseline scenario by adding more mobility, multi-hop TCP, and dynamic routing using AODV-UU. The UDP flow, again, is sent from node 2 to node 0. Node 3 now starts out alongside node 0 at position A, as illustrated in Figure 6. Node 2 starts its UDP flow to node 1 simultaneously as node 3 starts a TCP file transfer to node 0 and starts moving towards position D. After 62 seconds it will reach position D and then turn back and move towards node 0 again. During this time, the TCP flow to node 0 is sent over a path that increases from one hop to two and three hops, and reduced back to one hop as node 3 is on the way back. Note that in this scenario, TCP and UDP have competing data flows going over the same intermediate nodes.

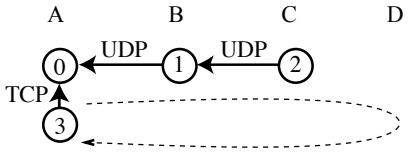


Figure 6. The Roaming Node scenario.

From separate analysis we know that there is one hop connectivity between node 3 and node 0 up to about time 25, followed by two hop connectivity up to time 50, and then three hop connectivity until time 92 when the route switches back to a one hop configuration until the end of the experiment. This is due to how AODV works; on the way back it will not optimize the route until a HELLO message is received by node 3 from the node 0.

The average UDP delivery ratio in this scenario decreased to 78.4% and the average TCP throughput to 0.93 Mbit/s [2]. A summary of the different scenarios in UDP deliver ratio and TCP throughput is shown in Table 2.

Figure 7 shows the UDP packet interspacing time and TCP sequence number graph for one of the experiments. It can be observed that the TCP data flow stalls in particular on the change from a one hop to a two hop configuration (time 25..30), and on the change from a two hop to a three hop configuration (time 40..50). There are also sporadic stalls during the three hop configuration (time 50..92). The different configurations are visible in the (slightly) decreasing slope of the time sequence graph. The switching of routes on the way back is much

[2] In the Roaming node scenario we experienced an outlier in one of our five test runs. Those results were therefore discarded and the average calculated over the remaining four runs.

Scenario	UDP Delivery Ratio	TCP Throughput
Static multi-hop UDP	99.2 %	-
Multi-hop UDP with TCP flow	97.4 %	1.34 Mbit/s
Roaming node	78.4 %	0.93 Mbit/s

Table 2. Average UDP delivery ratio and TCP throughput for the different scenarios.

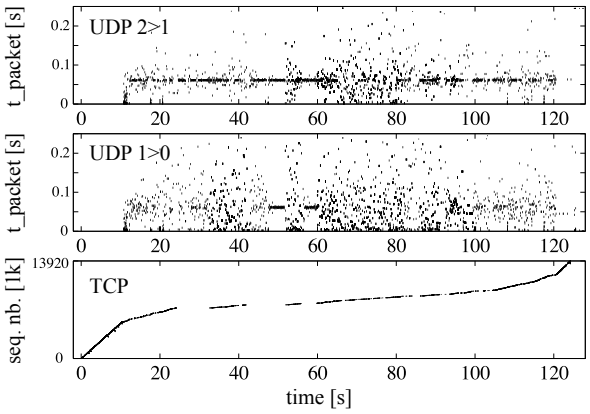


Figure 7. Example of UDP packet interspacing time and TCP sequence graph in the Roaming node scenario.

smoother. This can be explained by AODV's HELLO messages working more proactively on the way back, discovering a more optimal (shorter) route before the old one is gone.

We further observe increased UDP packet interspacing on the link between node 2 and 1 during three hop TCP connectivity (time 50..80). Increased UDP packet interspacing on the link between node 1 and node 0 is observed whenever we have progressing TCP traffic, but the UDP flow immediately stabilizes during TCP stalls. The increased packet interspacing is caused by the extra queuing delay on the intermediate nodes when the TCP flow also competes for buffer space.

An interesting observation can be made in the beginning of the experiment. It seems that when the TCP and UDP flows start simultaneously and in combination with dynamic AODV routing, the multi-hop UDP path between node 2 and 0 is broken. We see two different explanations for this (or likely a combination). Either TCP captures the channel, causing significant loss on the first hop UDP link (consequently no traffic is seen on the second hop link either). A more probable explanation, though, is that TCP's aggressive start impacts AODV's HELLO neighbor sensing and broadcast route discovery. The probing packets used in these mechanisms are broadcasted on the link layer, hence without acknowledgments or retransmissions. If these packets are lost, no path will be established. Not until time 10, after some movement will TCP back off and allow the UDP flow between node 2 and node 0 to resume. The reason for this is probably that node 2 was previously a hidden terminal for node 3, causing massive collisions at node 1. Only after node 3 moves within contention range of node 2, will it back off. The exact cause of this problem will be further examined in the next section, where we present a deeper analysis of protocol and link interactions.

3.5 Analyzing Protocol and Link Interactions with "Activity Plots"

The experimental analysis in the previous section raises many questions about the causes of the seemingly random and complex performance anomalies. It is necessary to study the interactions of protocols and link conditions at a much finer granularity in time to understand what is really happening. For this purpose we have developed an analysis tool to capture interactions between the protocols at different layers and for different nodes in a graphical way. This tool enables us to create "activity plots" from the collected data. Activity plots are a way to visualize and understand the timing of events. They complement the traditional flow based analysis with a more detailed analysis of activity on individual links. The idea is to present an experiment's complete set of events at a time granularity that is between single trace entries and aggregated performance figures.

An activity plot shows link activities on each nodes' wireless interface for all (or selected) links of a network in a single plot. A visual approach is better suited to comprehend the large amount of data collected during an experiment, providing for example an overview of the connectivity in the network and an easy way to spot connectivity problems. Coupled with traditional mea-

surements, such as a source node's TCP sequence number graph, it can be easier to identify and understand the spatial and temporal events that impact the performance.

We construct activity plots in the following way. During the experiment, all nodes record the successful reception of 802.11 frames, higher level events like route changes, and application layer specific data. The background time stamping application permits to synchronize the local traces after the test run and to merge the logs into a single experiment trace. From this trace file, we re-extract node-specific reception events and classify them according to their origin, creating a "link" view, and distinguish different message type in a graphical way. A link from node x to node y is denoted as $x \rightarrow y$. The links are laid out on the y-axis of the figure, while the x-axis represents time. Different activity events on a link are plotted as:

+ (large plus) unicast packet destined for node y.

+ **(small plus)** unicast packet overheard by node y.

◇ **(diamond)** broadcast packet. AODV route requests are highlighted with a large diamond.

We will now review some of our previous findings and discuss them using activity plots.

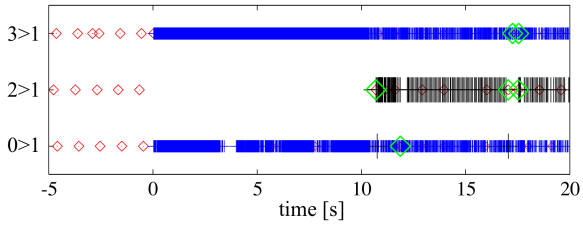


Figure 8. Activity plot showing the UDP flow starvation at the beginning of the experiment as seen by node 1.

An activity plot from the beginning of the Roaming node experiment from section 3.4 is shown in Figure 8. The plot shows all link activity as seen by node 1. In this scenario we know that node 3 is supposed to send TCP data to node 0 which is acknowledging the data packets starting at time 0. The activity plot shows that node 1 indeed overhears this traffic from nodes 3 and 0. At time 3, there seems to be a short interruption of the acknowledgments. A separate inspection of the link $3 \rightarrow 1$ however shows that these acknowledgments got to their destination (i.e., the TCP data flow is not interrupted).

The activity plot also shows that the UDP data flow sent from node 2 is not received until time 11. It further shows that the broadcast packets sent by node 2 (i.e., AODV HELLO messages), and received before time 0, stops being received up until time 11. From the activity plot we cannot conclude if node 2 was unsuccessful in contention for the wireless channel, or if node 1 did not receive the packets due to interference. If there would have been another node within connectivity of node 2 at that time, it might have overheard the packets. But from this information, we can not be sure. Clear though, is that the TCP flow affects the reception of HELLO messages. The route request visible at time 11 just before a burst of traffic on link $2 \rightarrow 1$, indicates that node 2 experienced problems to setup a route to node 1. The burst is caused by the transmission of buffered packets once a route is discovered. After the UDP flow is established, the packet interspacing of the overheard TCP data packets is increased. This is in line with the time-sequence diagram shown in Figure 7.

Another interesting part of the Roaming node experiment is the route change from one hop to two hops for the TCP flow around time 27. The activity plot in Figure 9 shows all activity originating from node 3, as perceived at node 0, 1, and 2 during the time when the route change occurs. A change in activity on the different links indicates connectivity problems on one of the links.

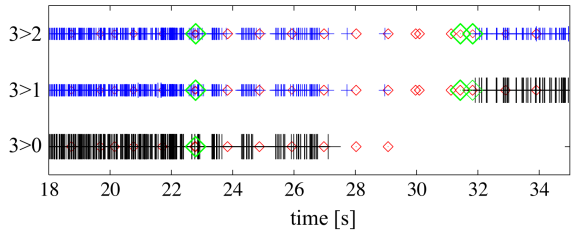


Figure 9. Activity plot showing all packets sent by node 3 illustrating the route change from one hop to two hops on the TCP link.

The route change going from a direct connection between node 3 and node 0, to a two hop route over the intermediate node 1, between time 27 and 32, can also be seen. The large plus symbols in Figure 9 indicate, before time 27, unicast packets on the link $3 \rightarrow 0$ and after time 32 on link $3 \rightarrow 1$.

Another interesting observation is that only broadcast packets are received by node 0 on the link $3 \rightarrow 0$ after time 27. After a while, also broadcast reception is impossible due to the increasing distance between nodes 3 and 0. The range where only broadcast but not unicast packets can be received is referred to as gray zones [6].

The main effect of the gray zone is on AODV, which during this time still assumes a valid route because it can receive broadcast HELLO messages, although no data traffic gets through. In the activity plot, it can be seen that nodes 1 and 2 actually overhear TCP retransmissions from node 3 while node 0 is in the gray zone of node 3. Only when broadcast reception stops will AODV time out its route. Not until after that time will TCP's retransmission trigger AODV to send a route request, searching for a new route (visible, for example, on the link $3 \rightarrow 1$ just before time 32). However, because of the exponential back off of TCP, this route discovery might not occur immediately after the gray zone. The TCP retransmission can be in a long timeout, effectively delaying the route re-discovery. This stall can be as long as twice the duration of traversing a gray zone, and severely worsens the effect of the problem.

4 Discussion and Conclusion

This paper has studied the effects on a low rate multihop UDP flow from a competing TCP flow. It was done through several experimental test scenarios where the TCP flow interfered, shared hops with the UDP flow and under mobility with dynamic routing. The purpose has been to study the interactions between the protocols at different layers.

The results indicate that although we use a moderate rate UDP flow, TCP's congestion control does not seem efficient enough to only have marginal impact on the other traffic in the network. When the two data flows do not share common links, we observe increased packet interspacing in the UDP flow, caused by jitter and to some extent packet loss. Instabilities in the form of short stalls are observed in TCP. Further analysis might show if TCP retransmissions are caused by lost packets or the high fluctuations in round trip time.

In the case where UDP and TCP share a common link, contention is significantly higher resulting in increased UDP packet loss and more significant TCP interruptions.

Dynamic routing, in particular when using broadcast neighbor sensing, adds another dimension of instability to ad hoc networking. Hidden terminals and channel capture effects cause otherwise stable routes to become unstable, simply because routing control messages are lost due to the competing data traffic. In our most complex scenario, the initial UDP flow experienced an average of 21.6% packet loss.

Our experiments show the complexity of the interactions in wireless multi-hop ad hoc networks. Therefore, we have developed an analysis tool to graphically analyze interactions between the protocols at different layers and for different nodes.

Acknowledgment

The authors would like to thank Laura M. Feeney for invaluable feedback on a prior version of this paper.

References

[1] The Uppsala University Ad Hoc Implementation Portal. http://core.it.uu.se/adhoc.

[2] G. Anastasi, E. Borgia, M. Conti, and E. Gregori. Wi-Fi in Ad Hoc Mode: A Measurement Study. In *Proceedings PerCom, Orlando (Florida)*, 2004.

[3] M. Gerla, K. Tang, and R. Bagrodia. TCP performance in wireless multi-hop networks. In *Proceedings of IEEE WMCSA'99 (to appear), (New Orleans, LA)*, February 1999.

[4] V. Gupta, S. V. Krishnamurthy, and M. Faloutsos. Improving the performance of TCP in the presence of interacting UDP flows in ad hoc networks. Unpublished.

[5] G. Holland and N. Vaidya. Analysis of TCP performance over mobile ad hoc networks. *Wireless Networks*, (8):275–288, 2002.

[6] H. Lundgren, E. Nordström, and C. Tschudin. Coping with communication gray zones in IEEE 802.11b based ad hoc networks. In *Proceedings of The Fifth ACM International Workshop On Wireless Mobile Multimedia (WoWMoM)*, September 2002.

[7] E. Nordström, P. Gunningberg, and H. Lundgren. A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In *Proceedings of Tridentcom*, February 2005.

[8] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP fairness in ad hoc wireless networks using neighborhood red. In *Proceedings of MobiCom'03, San Diego, USA*, September 2003.

[9] S. Xu and T. Saadawi. Does the ieee 802.11 mac protocol work well in multihop wireless ad hoc networks? *IEEE Communications Magazine*, 39(6), June 2001.

Hop of No Return: Practical Limitations of Wireless Multi-Hop Networking

Marina Petrova, Lili Wu, Matthias Wellens, Petri Mähönen
Aachen University, Department of Wireless Networks, Kackertstrasse 9, 52072 Aachen, Germany
E-mail: mpe@mobnets.rwth-aachen.de

Abstract

In this paper we report in a concise form our results from the measurement campaign that was performed to test throughput efficiency of multi-hop 802.11 networks, including also heterogeneous networks with 802.11a, .11b, .11g, and Bluetooth 1.1. One of the main contributions of the paper is that it is not reporting only simulations, but carefully monitored and calibrated measurements. The paper confirms the expectation that if relaying nodes have only one radio the number of wireless hops should be strictly limited.

1 Introduction

The wireless communication has emerged as a mainstream paradigm during the last decade. The number of wireless LANs, especially 802.11 based systems a.k.a. Wi-Fi, has increased extremely rapidly, and this development is, in part, driving the wireless *data* communications development.

It is a common practice to use simulation tools for analysing and verifying the behaviour of wireless networks as it is relatively fast way to obtain results. However, much remains to be done to make simulations more robust and reliable. The simulation based studies should also be validated with measurements at reasonable limits. In the case of WLANs, most of the constraints in the reliability come from the relatively simple PHY/MAC IEEE 802.11 implementation in many commonly used simulators [20]. Many earlier simulation studies have been focusing on testing the interaction between TCP and the wireless MAC protocols, evaluating the fairness and measuring the throughput analysis of TCP in multi-hop IEEE 802.11b scenarios, both in indoor and outdoor environment. Besides the numerous theoretical and simulation based articles several practical studies have been performed on wireless multi-hop networks [4, 10, 12, 18, 21].

The motivation of our work reported in this paper is to summarize the studies based on real testbed measurements that we made recently on wireless ad hoc networks. Several of them were done and reported about in other publications but doing a whole measurement campaign in one environment allows detailed analysis and comparison of results accepting that not all scenarios can be presented in full detail here. Great care was taken to find out practical problems that could be encountered with real physical deployment of wireless multi-hop mesh networks. In this paper we study the TCP and UDP performance in indoor IEEE 802.11b and heterogeneous multi-hop scenarios. The heterogeneous scenario combines devices with different wireless interfaces such as IEEE 802.11b/g and Bluetooth. Due to the limited range of the wireless interfaces used, multiple hops are typically needed to enable the communications.

Our target scenario is mostly based on the idea of mesh or community networks, i.e., most of the wireless nodes are static or very slowly moving. We present results from a comprehensive set of measurements and study issues such as TCP and UDP throughput and performance of the MANET routing protocols (AODV and OLSR) in our scenario. Since the users are mostly static (the topology of the network is not rapidly changing), we make some comparative studies using static and dynamic routing protocols. In addition, we also measure the difference in performance

between MANET routing protocols and OSPF (which is widely used in fixed networks and could be a possibility also for wireless environment as long as the topology is relatively static). This is motivated by the recent activity towards a new version of OSPF including extensions for wireless networks [14].

The rest of the paper is structured as follows. Section 2 gives a brief overview of the relevant work on wireless ad hoc networks outlining the constraints of TCP over wireless, the performance of the mostly used ad hoc routing protocols (AODV and OLSR) and the fairness and capture effect issues. In section 3 we give a short description of our testbed setups along with the comparison and analysis of the results. We also discuss the constraints and limitations of the present ad hoc networking solutions based on the measurement studies. We conclude the paper in section 4 by outlining some potential future research avenues.

2 Related Work

In a large number of recent studies on WLANs and ad hoc networks, authors have studied the performance of TCP over IEEE 802.11. The 'misbehaviour' of TCP over wireless is a consequence of several issues, and is well recognized problem [11, 13, 22].

First, as the topology of the ad hoc network can change and some of the wireless links can break, TCP will experience timeouts that lead to severe performance impact. TCP is a transport protocol that by changing its window size adapts the transmission rate to the available network bandwidth. As such, it functions successfully in the fixed networks. However, in the wireless network a packet can be lost not only due to congestion but also because of the errors in the wireless channel. Often link-layer contention in the case of hidden terminal problem can cause additional transmission errors. No matter the type of the loss, it is incorrectly interpreted as a sign of congestion, causing adaptation of the TCP window size and reduction of the data flow. Many efficient mechanisms have been proposed for improving the TCP performance in wireless ad hoc networks, see, for example, [6, 22] and references therein. Second, it is shown that TCP performance in an ad hoc multi-hop environment is sensitive to different parameters such as packet size and TCP window size [11]. It can also be verified from measurements that for a specific network topology and traffic flow, there is a TCP window size at which the throughput reaches the highest value. Further increase of the window size does not lead to a better result. This result is obvious, but it is not always properly taken into account. The third issue causing performance degradation is due to the interaction with the IEEE 802.11 MAC protocol. More specifically, when used in the multi-hop networks the present IEEE 802.11 MAC protocol may cause unfairness between competing TCP traffic flows, and a capture of the whole wireless channel by a single node can occur relatively easy.

Although the system could benefit from different protocol boosters or TCP modifications, we are leaving them strictly out from our study. We are limiting our measurement campaign to unmodified, off-the-shelf solutions, since these are building blocks that are mostly used in testbeds, community networks, and simulation studies.

The standard MANET routing protocols typically find routes using the minimum hop-count metric. Many studies have shown that this approach often selects paths with less capacity than the best possible paths existing in the network [10]. The reason behind that is the binary logic of the algorithm: in the process of choosing the paths it is assumed that the links are either able to deliver packets or not. This simple logic can be improved, if for example a link quality information (SNIR) is used as a criteria for selecting a high-quality link. An alternative solution is, e.g., expected transmission count metric (ETX), as suggested in [9].

Another point to mention is that many MANET routing protocols such as AODV, for example, do not take into account the effects caused by the multi-rate nature of the wireless standards. This implies an unreal assumption that all available links in the network are equal. In IEEE 802.11b for example the broadcasting of the HELLO messages is always done at 1 Mbps whereas the unicast data packets are sent at higher rates (up to 11 Mbps). Naturally the transmissions at lower bit rates, which in the case of WLANs correspond to different modulation and coding schemes, are more robust and can reach further. While the HELLO messages can be heard, the same does not apply for the data packets. This is a reason for appearance of the 'gray zones' [15], locations where the data pack-

ets experience a great loss. Furthermore Anastasi *et. al.* [4] discussed on the impact of different physical layer modes on the communication range, confirming what one would expect from theoretical PHY-layer analysis.

The number of portable devices with different build-in communication technologies is increasing rapidly. There are already several wireless technologies that can be used to build heterogeneous wireless networks. Naturally the necessity for interconnecting the devices using different technologies is becoming reality despite the obvious problems and performance constraints that such a network will face (e.g. reliable routing, higher delay because of forwarding between technologies, necessity for multiple interface node for bridging between technologies etc.). It is surprising how little heterogeneous and multi-radio environments are studied. Also the issue of harmonizing link-interfaces requires more attention. However, there are some activities on the domain of transparent link-layer technologies, such as Universal Link Layer API (ULLA) in GOLLUM-project [1], Generic Link Layer approach (GLL) as proposed by Ericsson, and earlier Wireless Adaptation Layer (WAL) suggestions.

3 Measurement Results

A common way to quickly estimate the performance of a specific wireless network is to measure the throughput. By throughput we mean the actual transport layer payload without any headers successfully received per second. In this section we shall analyze the TCP and UDP throughput in a homogeneous 802.11b multi-hop setup, we compare the throughput performance of 802.11a, 802.11b and 802.11g, and discuss the results from the channel allocation analysis and fairness issues when 802.11b and 802.11g are working together. We also give an overview of the outcome from the throughput measurements in a heterogeneous 802.11b, 802.11g and Bluetooth environment, and finally compare the performance of the AODV, OLSR and OSPF routing protocols.

The measurement setup in the multi-hop case was a simple string topology in office environment. All measurements were performed with laptops running Linux Kernel v2.4.26 and TCP NewReno with selective acknowledgement and enabled timestamps and RTS/CTS switched off.

3.1 TCP and UDP Measurements and Simulations in Multi-hop Environments

We shall begin with a 802.11b based multi-hop scenario where both TCP and UDP throughput are measured as a function of number of hops. Figure 1 shows the results of the multi-hop measurement campaign that was also used for setting up ns-2 simulations. Having in mind the shortcomings of ns-2, we improved the 802.11 MAC module and included enhanced error modelling. The reader is referred to [20] for further details. The measurements were taken using 802.11b WLAN PC-cards based on Intersil Prism2 chipset explicitly set to 11 Mbps PHY mode. Each measurement is averaged over several trials. Static routing was used in the measurements. On the other hand, simulations were based on AODV, but as there was no mobility the routing protocol did not have a great effect on the simulation results as one would expect. The reader should note that we adapted the propagation model in the simulation to be similar to the measurement environment.

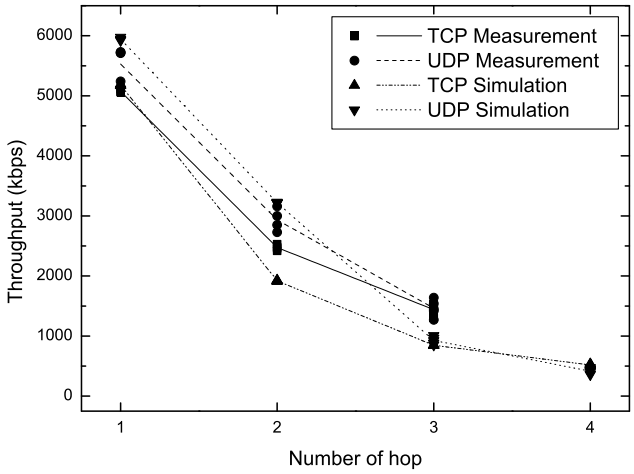


Figure 1. End-to-end TCP/UDP multi-hop throughput measurements and simulations.

The large number of wireless hops, in a single-radio per node case, is very inefficient, as throughput is lost rapidly. This is unavoidable even in the perfect envi-

ronment without transmission errors, delays, etc., as it is inherent for single radio repeaters. It is even more serious in the realistic Wi-Fi multi-hop environment. It is clear from figure 1 that already three hops are quite suboptimal for many purposes. Further increasing of the number of hops will result in even lower throughput. In our tests there were no external nodes contending for the channel. However, at a public hot spot other users will also produce interference, so the end throughput would fluctuate more and experience further degradation.

Tschudin *et. al.* have also introduced the *ad hoc horizon* [19] of about three hops for 802.11 networks and TCP. Our measurements confirm their simulation-based results and extend their setup by taking into account UDP transmissions and higher data rates.

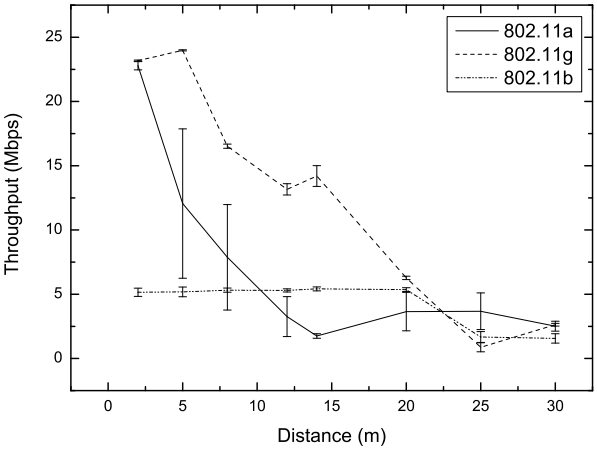


Figure 2. TCP Throughput measured with 802.11a, 802.11b and 802.11g.

3.2 Comparing IEEE 802.11a, .11b and .11g

After measuring the performance of IEEE 802.11b we also included 802.11a and 802.11g devices into the testbed. Both technologies are based on Orthogonal Frequency Division Multiplexing (OFDM) modulation scheme in 5 GHz and 2.4 GHz frequency band and support bit rates from 6 to 54 Mbps.

There has been some uncertainty on how 802.11a compares to 802.11g. Both technologies are quite similar on PHY layer but use different frequencies. Naturally the radio implementations are at least slightly different. In order to compare the performance of both technologies in a real indoor environment, a set of single-hop throughput measurements was performed. For each technology, the end-to-end TCP throughput was measured at 8 different points (range from 2 m to 30 m) using a packet size of 1500 bytes. In addition to 802.11g (Prism GT chipsets) and 802.11a (Atheros AR5211 chipsets) we made 802.11b measurements (Prism2 chipsets) for comparison.

Figure 2 shows the TCP throughput of 802.11a, 802.11b, and 802.11g as a function of distance. In general this graph also shows the SNR-dependency of the TCP-throughput since they are closely related. Comparing the 802.11a and 802.11g curves, we can identify three segments: At short distances with LoS (line of sight), as expected, both technologies reach the same maximum throughput of about 23 Mbps. When the nodes are further away from each other ($>$ 20 m), with obstacles in between, both technologies switch to the lowest bit rate to maintain the connection. In the range in between, the throughput of 802.11g clearly outperforms 802.11a. Due to the higher path loss of 802.11a, the physical layer mode was already switched to more robust modulation and coding that led to lower bit rate. On the other hand, the throughput of 802.11b is rather stable. At some points (e.g. distance from 12 m to 20 m), 802.11b even outperforms 802.11a cards. In the whole measurements the effect of the rate adaptation is clearly visible.

It should be mentioned that the performance of the devices depends on the deployed hardware. Wireless LAN cards from different vendors, such as described in [5], can have a distinct impact on the data throughput, although these differences will not change the general trends.

In general, the measurements show that 802.11g and 802.11b can benefit from smaller propagation path loss, and consequently 802.11g has larger range than 802.11a at the same bit rate. The OFDM-modulation is only beneficial for rather small distances where the SNIR at the receiver is still high enough.

3.3 Impact of Channel Assignment on 802.11b

In case the density of WLAN hot spots is high, the channel selection becomes an important issue. We

address this problem by comparing a set of measurements done with different channel assignments for two simultaneously running traffic flows. The hosts are located close to each other. Two TCP transmissions (the packet size is 1500 Bytes) run simultaneously for 20 s, and the average throughput is recorded. In each measurement we changed the channel assignment, so that the channel distance between the two transmissions corresponds to 0 (same channel) up to 5, having in mind that there are 11 channels available in the 802.11b standard. Note that the difference between the channels is not in frequency domain but according to their numbering from 1 to 11. Figure 3 clearly shows the dependency of the throughput on the channel selection. When both transmissions run in the same channel, the total throughput is close to the maximum throughput for a single transmission. This indicates the CSMA scheme of 802.11 MAC can fairly resolve the competition in the same channel without decreasing channel capacity. Since the frame collisions in the neighbouring channels (1 or 2 channels apart) cannot be effectively avoided by CSMA the total throughput is lower than in the single channel case. When the frequencies are assigned 4 channels apart the throughput is almost the same as in the non-overlapping channels case. The results show that neighbouring channels should be avoided when assigning frequencies although slightly overlapping channels could be used in a dense environment. We have earlier proposed a distributed channel assignment mechanism based on a graph colouring algorithm that minimizes the interference and improves the performance [16].

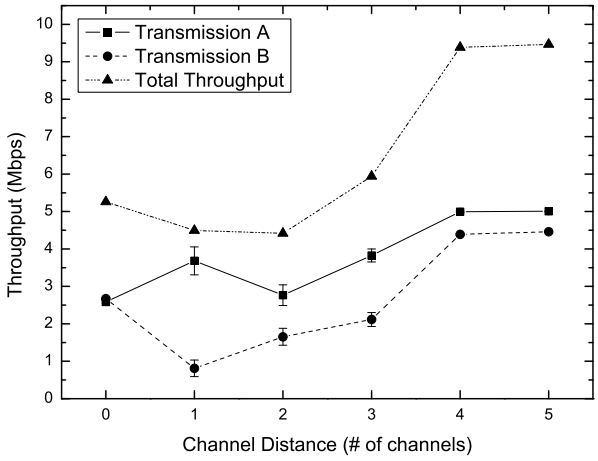


Figure 3. Comparison of different channel allocation schemes with 802.11b.

3.4 Contention between IEEE 802.11b and IEEE 802.11g

IEEE 802.11g was designed to offer higher datarates still keeping the backward compatibility to the most popular IEEE 802.11b. Therefore it also works in the 2.4 GHz ISM-band and supports the basic CCK-modulation of 802.11b.

The interworking between well-known 802.11b devices and nodes using 802.11g PC-cards is an interesting research issue. Therefore we analyzed the contention between one 802.11b link and another 802.11g link, both running simultaneously.

Figures 4 and 5 show both the TCP and UDP throughput measured at each single link as well as the aggregate throughput of the two flows. As it can be seen from figure 4, we started the second transmission five seconds later. The result will not change if we reverse the order of the transmissions. By comparing the averaged throughput results during the contention phase and the first or last five seconds one can calculate that the 802.11g reaches $\sim 48\%$ of its maximum TCP throughput. In contrast 802.11b reaches only $\sim 26\%$ of its maximum. The same capture effect can be determined in the case of UDP traffic. As UDP does not send acknowledgements the collision probability of one transmission is lower so that the percentages are higher (802.11g: $\sim 67\%$ and 802.11b: $\sim 40\%$) but the capture-effect can still be seen.

If both connections are not anymore configured to the same channel but adjacent channels the general result will not change. As nodes working on different frequencies do not sense each other's transmissions frame collisions become more probable and the total throughput decreases. The unfair effect of 802.11g capturing the channel is less severe but still present.

As 802.11g is based on OFDM modulation it is in principle more robust against transmission errors. Furthermore the receivers are more sophisticated compared to the older 802.11b devices so that there is a higher probability for the 802.11g interfaces to re-

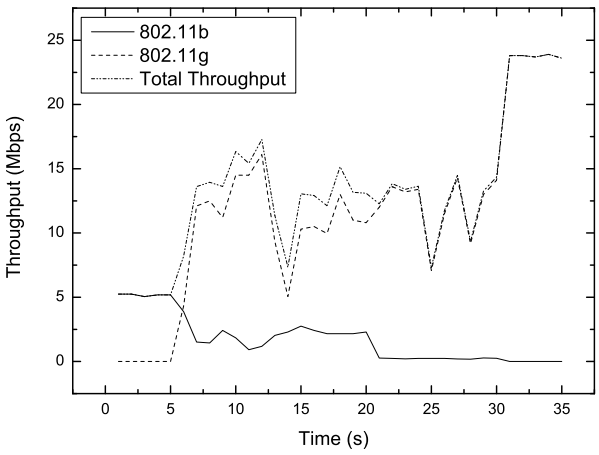


Figure 4. Contention between 802.11b and 802.11g for TCP traffic.

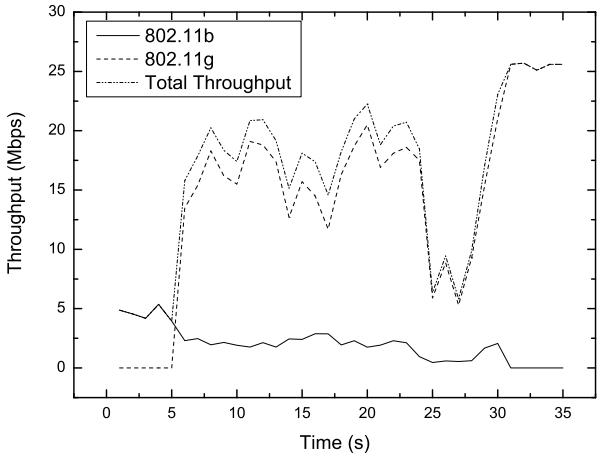


Figure 5. Contention between 802.11b and 802.11g for UDP traffic.

ceive a frame although a frame collision occurs. The 802.11b may loose its transmission and therefore defers for an extended interframe space and doubles its backoff window. The higher frame loss and the resulting higher average backoff period cause the smaller 802.11b throughput. As this capture effect depends on the sensitivity of the device it may look different if other WLAN cards are deployed.

3.5 Heterogeneous technology scenario

For the heterogeneous scenario we extended our measurement setup by using *Bluetooth* as another wireless technology working in the 2.4 GHz ISM-band. After measuring the single hop performance we combined it with 802.11g and 802.11b to form a heterogeneous 3-hop connection. We used static routing such as described for the homogeneous multi-hop measurements and set the two WLAN links to different channels to avoid interference between them. The interference between BT and WLAN was not avoidable as we deployed devices based on BT v1.1 that do not adapt their hopping sequences in order not to interfere frequencies used by nearby WLANs.

For detailed analysis of the interference effects between BT and WLANs the reader is referred to [7].

We again generated TCP traffic using Iperf and measured the end-to-end throughput. Figure 6 directly compares the heterogeneous results to the single BT link. Although we extended the connection by adding an 802.11g as well as an 802.11b link the maximum throughput only decreases from 395 kbps to 360 kbps. As expected, the slow BT link forms a bottleneck for the whole transmission and the performance of the end-to-end connection is only determined by this bottleneck-link. We verified this behaviour using a heterogeneous link consisting only of 802.11b and 802.11g that reached nearly the same performance as a single 802.11b link. If both WLAN-links were using non-overlapping channels the throughput would be as fast as a single 802.11b link. However, if both links were using the same channel the transmission would not be much slower.

The heterogeneous multi-hop connections are more or less limited to the performance of the slowest link involved. This behaviour limits the usability and the applications of such networks. However, having heterogeneous connections in the network can be used to, e.g., extend the range of BT or interconnect technologies.

3.6 Comparison of Routing Protocols

We used a quite simple three-hop string topology as presented in subsection 3.1 to study routing protocol sensitivity. In order to introduce alternative paths

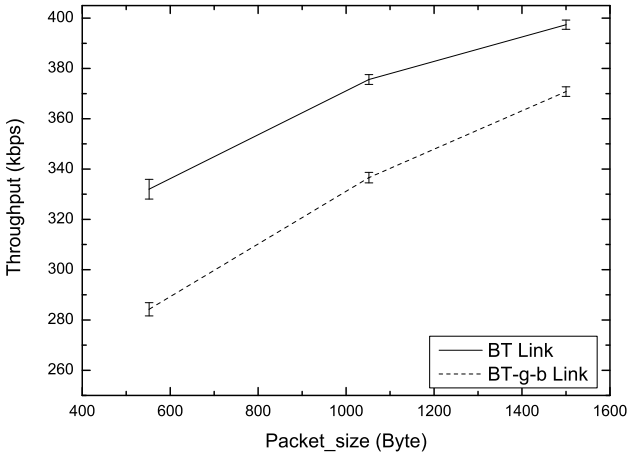


Figure 6. Comparison of a single BT link and a heterogeneous 3-hop connection consisting of BT, 802.11g and 802.11b.

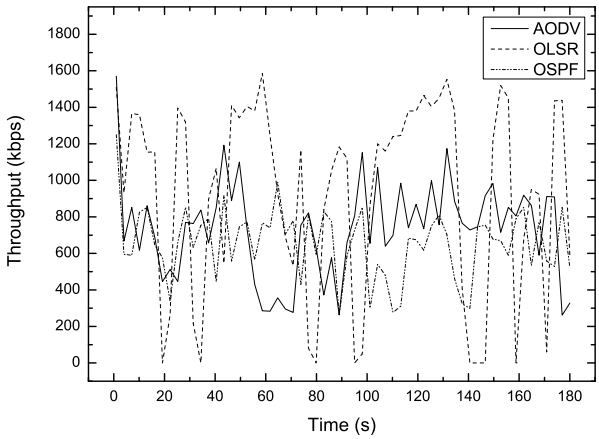


Figure 7. Comparison of different routing protocols: AODV, OLSR and OSPF.

we extended the topology by adding a second parallel node at both intermediate hops. The final static scenario consists of 6 nodes using 802.11b interfaces.

Besides the well-known ad hoc routing protocols AODV [17] and OLSR [8] representing the reactive and proactive approach, we chose OSPF as a third routing protocol. Since we were using a static topology we expected a relatively stable scenario and OSPF is an interesting option because of its wide deployment in the fixed networks enabling a much easier integration of existing wired infrastructure and new wireless extensions.

We measured the TCP end-to-end throughput with packet sizes of 1500 bytes. Figure 7 shows the measured throughput over a period of 180 s.

Although we used a static topology all routing protocols reveal stability problems; OLSR even looses the connection for several seconds. This leads to a smaller average throughput compared to the measurements with static routing presented in subsection 3.1.

When comparing the three protocols in detail OSPF performs worst. The unreliable nature of the wireless channels is the main reason as OSPF was not designed for such environments. However, the performance is not much worse and no connection breaks occurred. The extended version of OSPF may be an interesting option for future work. The used OLSR implementation from UniK (Universitetsstudiene på Kjeller) [2] reaches for several periods the maximum possible throughput but suffers severe instability problems. The AODV implementation from Uppsala University [3] on the other hand does not reach the same maximum throughput but is more stable overall. We cannot draw a final conclusion of which protocol is the best suited for static small-scale scenarios as none reached sufficient stability, which is one of the main desirable requirements for routing protocol in possible commercial deployments.

4 Conclusions

The results from our experimental study show that massive scale multi-hop networks based on unmodified 802.11 and TCP are not efficient. They can be build, but there should be clear reason for paying the price of considerably lower performance. In many consumer and civilian applications the justification might be difficult to find. However, in the limiting small-scale use one could consider the possibility of applying multi-hop techniques for extending the range of an access point in infrastructure mode. Adding one or two extra hops between the access point and the end user device reduces the need for infrastructure and increases the flexibility of the network.

We would also like to stress that although the WLANs are quite plug-and-play for single-hop consumer use, when one is building optimized multihop networks this is not the case. Especially when one is programming lower level functionalities or tries to read MAC/PHY-layer parameters the lack of well documented interfaces makes it difficult. Besides, MANET routing protocols do not reach sufficient stability even in small-scale static scenarios.

As WLANs become more widespread and heterogeneous (802.11b and 802.11g are today's commercial standard)the shown capture effect of more modern transceiver hardware and the smart frequency allocation will become issues that require more serious considerations when deploying these networks.

If one is considering strictly the present day technology without extensions, we recommend staying below four hops in order to keep system performance at the reasonable level. As pointed out also by other authors, if one wants to build mobile and massive ad hoc networks, unmodified 802.11 technology might be inadequate - at least in the case of single radio per host.

5 Acknowledgments

We would like to thank the European Union (6HOP, MAGNET and GOLLUM-project) for providing partial funding for our work.

References

[1] *The GOLLUM-project website*, http://www.ist-gollum.org [Cited on: 27th April 2005], 2004.

[2] *UniK OLSR Implementation*, http://www.olsr.org [Cited on: 19th April 2005], 2004.

[3] *Uppsala University AODV Implementation*, http://core.it.uu.se/AdHoc/AodvUUImpl [Cited on: 19th April 2005], 2004.

[4] G. Anastasi, E. Borgia, M. Conti, and E. Gregori. Wi-Fi in Ad Hoc Mode: A Measurement Study. In *Proc. of PERCOM'04*, March 2004.

[5] Atheros Communications. *White Paper: 802.11 Wireless LAN Performance*, USA, 2003.

[6] B. Bakshi, P. Krishna, N. Vaidya, and D. Pradhan. Improving Performance of TCP over Wireless Networks. *Technical Report 96-014, Texas A&M University*, 1996.

[7] C. Chiasserini and R. Rao. Performance of IEEE 802.11 WLANs in a Bluetooth Environment. In *Proc. of WCNC*, Sept. 2000.

[8] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized Link State Routing Protocol. In *Proc. of INMIC '01*, 2001.

[9] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of MobiCom '03*, 2003.

[10] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris. Performance of Multihop Wireless Networks: Shortest Path is Not Enough. In *Proc. of HotNets-I*, Oct. 2002.

[11] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The Impact of Multihop Wireless Channel on TCP Throughput and Loss. In *Proc. of INFOCOM '03*, March 2003.

[12] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang. TCP over Wireless Multihop Protocols: Simulation and Experiments. In *Proc. of ICC '99*.

[13] A. Gurtov and S. Floyd. Modeling wireless links for transport protocols. *ACM SIGCOMM Computer Communication Review*, 34(2):85–96, Apr. 2004.

[14] J. Ahrenholz, T. Henderson, P. Spagnolo, E. Baccelli, T. Clausen, and P. Jacquet. *OSPFv2 Wireless Interface Type, IETF draft, expires Nov. 2005 (work-in-progress)*, 2004.

[15] H. Lundgren, E. Nordström, and C. Tschudin. Coping with communication gray zones in IEEE 802.11b based ad hoc networks. In *Proc. of WoWMoM '02*, 2002.

[16] P. Mähönen, J. Riihijärvi, and M. Petrova. Automatic Channel Allocation for Small Wireless Local Area Networks using Graph Colouring Algorithm. In *Proc. of PIMRC 2004*, Sept. 2004.

[17] C. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proc. of WMCSA '99*, 1999.

[18] L. Qiu, R. Chandra, K. Jain, and M. Mahdian. Optimizing the Placement of Integration Points in Multihop Wireless Networks. In *Proc. of ICNP '04*, 2004.

[19] C. Tschudin and E. Osipov. Estimating the Ad Hoc Horizon for TCP over IEEE 802.11 Networks. In *Proc. of Med-Hoc-Net '04*, June 2004.

[20] M. Wellens, M. Petrova, J. Riihijärvi, and P. Mähönen. Building a Better Wireless Mousetrap: Need for More Realism in Simulations. In *Proc. of WONS '05*, 2005.

[21] K. Xu, S. Bae, S. Lee, and M. Gerla. TCP Behaviour across Multihop Wireless Networks and the Wired Internet. In *Proc. of WoWMoM '02*, 2002.

[22] X. Xylomenos, G. C. Polyzos, P. Mähönen, and M. Saaranen. TCP Performance Issues over Wireless Links. *IEEE Communication Magazine*, Apr. 2001.

SESSION III

Deploying MANET Test-beds

Thoughts on Mobile Ad-hoc Network Testbeds

Wolfgang Kieß, Stephan Zalewski, Andreas Tarp, Martin Mauve
Heinrich-Heine University Düsseldorf
{kiess,mauve}@cs.uni-duesseldorf.de
{stephan.zalewski, andreas.tarp}@uni-duesseldorf.de

Abstract

Currently, several research groups seek to develop appropriate tools and methodologies for real-world experiments on mobile ad-hoc networks. We argue that this should be done as a community effort rather than as independent projects. Furthermore we present our view on the functionality a good testbed should provide based on reports from other research groups as well as on our own experiences. This paper aims at stimulating a discussion, it is not meant to be a comprehensive specification of requirements or a detailed workplan.

1 Introduction

The two most common instruments used to evaluate algorithms for mobile ad-hoc networks (MANETs) are theoretical analysis and network simulation. Theoretical analysis provides fundamental insights into the characteristics of the investigated approaches, simulation enables their exploration in a dynamic environment. Both methods require a significant level of abstraction to reduce the real-world complexity of mobility, radio propagation, and hardware. As it has been discussed in [13], the direct transfer of findings from simulations to real-world systems is thus not advisable. As a consequence, theoretical analysis and simulation should be complemented by real-world experiments.

A surprisingly large number of these real-world experiments have already been conducted, leading to results and insights that were not foreseen in simulation or theoretical analysis. Most of the experimenters also had to face unforeseen difficulties. Nodes failed during experiments which was only discovered after the experiment, results were difficult to reproduce and therefore work was unnecessarily duplicated by multiple work groups or the network showed unexplainable behavior. We believe that these problems can be alleviated if future experiments satisfy three key requirements of scientific experimentation:

- **Reproducibility.** Independent research groups must be able to reproduce the results of an experiment. For mobile ad-hoc networks, reproducibility is a significant challenge due to the complex impact of radio propagation and node mobility on the results of an experiment.

- **Comprehension.** A scientist conducting an experiment must be able to access all relevant information to comprehend and explain the results of the experiment. There is a need for tools that collect information on different layers and combine this information to allow a detailed analysis.

- **Correctness.** Any experiment may suffer from broken tools, errors with the setup and problems when conducting the experiment. While reproducibility and comprehension will most likely reveal these problems, it is vital to the efficiency of a researcher to be able to verify whether any given experiment has produced valid results. This can be supported by an established methodology and a selection of appropriate tools.

To address these issues, there are efforts underway that aim at the development of methodology and testbeds to support systematic experiments in mobile ad-hoc networks. One early example is the APE

project [14], introducing, among other things, virtual metrics for node distance in order to increase reproducibility. Several other testbeds are currently being worked on [19, 8]. All these efforts provide very valuable individual contributions. However, it is our belief that a community effort is necessary to establish real-world experiments as the third standard instrument to evaluate algorithms and protocols for MANETs. This effort must include an analysis of the functionality expected from a "perfect" testbed as well as a discussion of the methodology for conducting experiments. Ideally this will lead to the design of a common architecture where the individual research groups can contribute reusable building blocks for a testbed that is supported by the ad-hoc networking community.

This paper is meant as our initial contribution to this effort. It describes work in progress not final results. In Section Two we present our view on a perfect testbed for mobile ad-hoc networks. Section Three describes some of our experiences with experimenting and Section Four concludes the paper.

2 The Perfect Testbed

In the following we assume that a testbed consists of two key elements: a number of physical devices (nodes) which may be moved around individually and the software to support and conduct the experiments.

In general an experiment for MANETs can be divided in several phases: implementation, experiment specification, node configuration, setup verification, execution, and analysis. For each of these phases we discuss how an ideal testbed could support them and present the current state of the art in this area.

2.1 Implementation

The first phase of an experiment is the implementation of the algorithm to be tested. A good testbed will support this phase in three ways: it will 1) help to minimize the work required for the implementation; 2) seek to reduce implementation errors; 3) encourage interoperability between algorithms implemented and evaluated by distinct research groups.

As a lot of algorithms will be initially analyzed by means of simulation, reusing the simulation code instead of reimplementing it eases the workload and reduces the potential for errors. Thus tools allowing to use the same code both for simulation and the real experiment are a vital component of a MANET testbed. Additionally, the testbed should support the implementation by providing well tested libraries for common MANET functionalities such as beaconing or for tasks such as tracing.

Encouraging interoperability is mainly a matter of interfaces and methodology. A good testbed will specify concise interfaces and best-practice methods for integrating new functionality such that it can be reused by other research groups. It will also support interoperability through a clean and simple architecture.

The concept of reusing simulation code for real-world experiments has already been utilized in some projects, namely in SURAN [1], WINGS [5] and the routing protocol evaluation presented in [23]. A particularly promising approach seems to be nsclick [16] which allows to run a click router [12] inside the ns-2 [17] network simulator.

There also exist libraries supporting the implementation of algorithms for ad-hoc networks. One class are libraries such as the PICA API [2], designed to provide platform-independency for the implementations using them. Other libraries offer additional functionality like neighbor discovery, flooding or packet buffering during route discovery. One example is the ad-hoc support library [11].

2.2 Experiment Specification

After the implementation is complete, the experimenter specifies the scenario used for the evaluation. In order to allow other research groups to verify the results, the specification should be a complete description of the experiment made available as a file in a standardized format.

There exist at least two variants of scenarios, *strict* and *loose scenarios*. In a strict scenario each node follows detailed instructions on when to perform which action. Although a rigid description of the experiment fosters repeatability, there are setups in which this is not suitable, e.g., if experiments are run as background tasks on devices primarily used for other purposes. In this case, a scenario with loose descriptions of the services and actions able to adapt to the current state of the node is better suited. In both cases a good testbed

will support the specification of an experiment through a set of tools. At the very least these tools will allow the planning of node mobility and the timing of events via a graphical user interface.

The only widely known tool for the specification of strict scenarios are the choreography descriptions for the APE testbed [14]. To our knowledge there are no tools that support the specification of loose scenarios.

2.3 Node Configuration

When the scenario is prepared, the nodes need to be configured with the information required to run the experiment. This includes the implementation of the investigated algorithms as well as the specification of the actions and the movements of each node. This step mainly comprises the distribution of files and the configuration of nodes (e.g., setting of addresses), thus it should be automated as much as possible.

The key to the autonomous configuration of the nodes is the experiment specification. Since this specification contains any relevant information on how each node should behave, a good testbed will be able to install the required software and perform the necessary configuration based solely on this information. This can either be done by directly distributing the specification to each node or it can require its "compilation" to gain configuration files that are specific for each node.

If the nodes are physically accessible to the experimenter, the automatic file distribution can be provided with simple means, e.g., through a one-hop download. However, in loose scenarios the devices are normally not available for a direct download and the required files therefore have to be distributed to nodes that are already in the field. One approach to do this is to let the nodes distribute the required files amongst themselves, i.e., whenever two nodes come in radio range, they will exchange information and files on the scheduled experiments.

Automatic configuration is a concept used in APE [14], the PRNET project [9] provided a mechanism for the remote configuration of nodes.

2.4 Verification of the Setup

The actual execution of an experiment that uses a strict scenario is extremely costly in terms of manpower and time. A verification of the test setup and the used hardware in the forefield of the experiment in a controlled laboratory environment is therefore vital and should be supported by the testbed.

The verification can be divided into tests involving one or multiple devices. Single device tests allow to avoid problems such as lack of memory, low battery power or physical damages. Tests with multiple devices can reveal problems only occuring due to the interaction between devices. An important multiple device test which should always precede an experiment is running the complete setup installed on real devices under laboratory conditions. Although these artificial conditions prevent the acquisition of quantitative results, the setup is not expensive, can be repeated easily, and allows the isolation of errors.

There exist a number of approaches on how to create a multi-hop network topology for the in-lab verification, e.g signal attenuation [10] or *MAC filters* that drop packets based on the MAC source address of packets and the virtual position of the sender [15, 22]. A good testbed can use the position information in the scenario file to compute the virtual distances and control the topology accordingly.

2.5 Support during the Experiment

The main phase of the experiment starts with the distribution of the devices. Each experiment will most likely consist of several runs in which the nodes move around. Finally, the devices need to be collected and the phase is concluded by collecting the tracefiles from the devices. The main phase has some properties which necessitate a dedicated support by the testbed: 1) The time in this phase is expensive, only an optimal usage of the experimental time makes experiments economically feasible; 2) Repeatability of this phase is crucial for a scientific evaluation; 3) All information available here is valuable; Therefore, the testbed should support the experiment by optimizing the usage of experimental time, by guaranteeing, or at least fostering, repeatability and by collecting detailed information on the nodes' actions.

The usage of the experimental time can be optimized by automating tasks and by avoiding errors and therefore unnecessary repetitions. As device distribution and collection are physical tasks, the potential

for automation is small, here. This is different with tasks not requiring a direct (human) interaction like the tracefile collection or the movement of the nodes which can be automated by mounting the nodes on robots [3]. A large optimization potential also lies in the avoidance of the execution of erroneous experiments. By controlling that all nodes act within the parameters specified in the scenario, the testbed should assure that exactly the intended experiment is executed.

The repeatability of an experiment is provided if it is possible to rerun the same experiment such that the relevant parameters in both runs have sufficiently similar values. There are two ways to support repeatable experiments, either by comparing the parameters after an experiment to determine if it was a repetition of a prior experiment or by steering the experiments to ensure that these parameters lie within an acceptable threshold. Open issues in this context are the determination of the relevant parameters and the question if it is technically and economically possible to record these parameters.

For all aspects mentioned so far it is crucial to trace the data on the behavior of nodes and on external influences as completely as possible. This data can be used for a detailed post-run analysis as well as for the steering of the experiment. The data to be recorded involves packet-level traces, timing and positioning information, states of higher level protocols as well as physical and MAC layer logging.

To steer the experiment, the *experiment control* component of the testbed should continuously compare the actual values of the relevant parameters to those specified in the scenario. The testbed therefore should provide a method to specify and control boundaries for these parameters, soft boundaries like "position between x-5 and x+5" as well as hard boundaries like "GPS daemon running". In case some of these boundaries are violated, the testbed can adjust the behavior of the node during the run or mark the run as invalid. If the violation is severe, this can render the whole experiment unusable and should therefore be known right during the experiment. Thus, the testbed should support the transmission of status information to a central monitor station and it should also be possbile to remotely login to the affected node to correct errors or alter the configuration.

The experiment monitoring and the remote login necessitate the exchange of management messages between the nodes and the monitoring station, possibly during the experiment. This counteracts a primary design goal of the testbed, i.e., minimizing the interference of the test equipment (both hard- and software) with the experiment. A solution to this is the "out-of-band"-transmission of all management messages. This can be achieved either via a separate network interfaces during the experiment, as payload of experiment packets with a dummy payload or in the pauses between the single runs of an experiment using the tested network itself. Although the last method does not allow to stop erroneous runs directly, this is not problematic if runs are short. Furthermore, it is more practical than the other methods as the first method is more expensive and the second not always feasible.

The concept of monitoring the state of the network during the experiment has been used in the PRNET Network Monitoring [9], the SURAN Automated Network Manager [1], the CMU Position and Communication Tracking daemon [15] and ATMA [18]. For some of the tracing tasks, there exist standard tools like tcpdump [21] for packet level tracing or gpsd [6] which can help to record the node position. A means to synchronize the clocks of the nodes is the usage of a combination of NTP and GPS. The concept of determining if an experiment is a repetition of another one by hindsight has been used in the APE virtual mobility [14]. To our knowledge, there exists no experiment control that adapts the nodes' behavior dynamically to changes in the environment to foster repeatability.

2.6 Postprocessing of the Experiment

The postprocessing can be divided into organizing the raw data gathered during the experiment and analysing it. This phase should be governed by the principle that the raw data is a valuable resource. It needs to be documented, stored and published. Based on this data, independent researchers must be able to verify any conclusions that are drawn from the experiments. A good testbed will provide mechanisms to ease the documentation, storage and publication of the involved files. Furthermore it will provide or incorporate an extensible toolset for analysing the raw data.

The first postprocessing step is the structured, permanent storage of all raw data. As there are also a lot of other files involved in the postprocessing such as scenario files or tools, the testbed's automatic file handling should include these. One possibility is the implementation of a file management framework that defines interfaces to access, view, annotate, process, and store these files. The organization of the raw data is concluded by the documentation of the events and conditions not recorded in the traces but perceived by the human participants.

The tools used to process the raw data provide functionality for consistency checks, data analysis or for enabling trace-based simulation. All these tools can be used for multiple experiments, thus the testbed should foster reusability to reduce work. If the tools are modular and reusable, this also increases reliability as results can be easily reproduced. Therefore the goal has to be the creation of a reusable standard toolset for the postprocessing of MANET experiments which should be publically available, extensible and well documented. The analysis tools should support the different input formats of real-world traces and simulator traces as this allows a direct comparison of results from both evaluation methods.

The file management as well as the toolset can be combined with a server that is publically available. If software tools and raw data are available in an open repository, other researchers can access and reuse the data and tools or verify the results. We are not aware of a toolset or testbed that supports the properties described above.

2.7 Orthogonal Concepts

The previous sections show how a testbed can support the single steps of an experiment. Besides that, there are general concepts such as error avoidance, reduction of workload, portability, modularity and benchmarking not limited to single steps.

Benchmarks provide references to improve comparability. One option to realize benchmarks for MANETs are *baseline protocols*. A baseline protocol must 1) have the best performance in its class, typically achieved through the use of "illegal" means such as global knowledge; 2) be easy to implement; 3) be easy to test; If the baseline protocol is included in the experiment, it builds an upper bound for the performance of the whole class of algorithms in this special setting. Instead of giving absolute numbers for the performance of an evaluated protocol, it can be expressed as percentage of the maximum. Candidates for baseline routing protocols are MERIT [4] or the "best-case" routing described in [22] which both use global knowledge to make routing decisions.

Another form of benchmarks are *standard tests* and *standard measure values*. Standard tests can be standard topologies such as chains or grids as well as tests on the maximum load the network can support or the load without data traffic. Standard measure values provide characteristics of a protocol. Examples are delay, packet order, route length or loss rate.

To reduce the workload and the errors arising due to reimplementing the same code multiple times, using the same source code for simulation, emulation, and real-world experiments is highly desirable. We call this SER integration. An additional advantage of this approach is the feedback that can be given between simulation and experiment. SER integration should be used for the evaluated algorithms as well as for scenario files and analysis tools. This means that the same scenario file should be able to drive a simulation as well as an experiment or that the output of both steps can be processed with the same analysis script.

Portability of the testbed software is very important since a test setup used for large MANET experiments will consist of heterogenous devices. As the full number of devices is only needed for a short time, sharing devices among workgroups and using devices like laptops or cell phones not especially bought for experimenting will be common in experiments. Furthermore not all workgroups will buy the same devices, leading to further heterogeneity. Apart from this, the product live cycle of mobile devices is short. If a device is bought today, it may be not possible to buy the same product in a year. Thus, the testbed software must be very portable to run on these different platforms.

Finally, it seems beneficial to employ a highly modular testbed architecture rather than a monolithic approach. This mainly is due to the large selection of tools allready available, including standard software such as tcpdump [21] or gpsd [6]. These tools should be used rather than reimplemented. In addition modularity will allow the easy exchange of components to

enable a competition on the level of individual components rather than complete testbeds. As a consequence we believe that the system should be only loosely coupled with some kind of "glue" combining these components to form the testbed. An example where this concept has been used to some extend are the scripts controlling APE [14].

3 Experiences

Since 2002 we have conducted experiments on real-world vehicular ad-hoc networks within the context of the FleetNet project [7]. The experiences gained through these experiments and the problems we encountered motivated us to investigate how to improve experimentation with real-world implementations of ad-hoc networks in general.

In a first step we tried to repeat experiments conducted by other research groups to confirm their results. This turned out to be extraordinarily difficult since there is almost always insufficient information available to reproduce the described results. Key issues were the lack of information on the environment (in particular regarding the setup of the experiment, the radio characteristics and the connectivity between nodes), no access to the raw data gathered during the experiment and a vast heterogeneity in the tools to setup, conduct and evaluate an experiment. These experiences led us to the first attributes of a good testbed: all information, code and tools need to be published, preferably in a standardized way.

We have conducted in- and outdoor experiments of a simple flooding algorithm for static ad-hoc networks with seven to ten nodes. The motivation for this very simple setup was to isolate problems that would lead us to general design criteria for experiments with mobile ad-hoc networks. In the following we sketch the main observations, more details can be found in [20].

Inspired by [15], we started by measuring the radio ranges of our hardware (iPAQ5550 IEEE 802.11b) and discovered that the iPAQ radios were sometimes able to successfully deliver ping packets over more than 900 m while already a tree in the line-of-sight between two nodes can block a transmission. Thus, setting up a reliable, reconstructible 7-node/6-hop string topology for preparatory tests was only possible by carefully positioning each device around the edges of a building.

Our next step was to set up a multi-hop topology where every node had multiple neighbors. After several measurement sessions, a suitable experimental site seemed to be the university parking lot. Later on we discovered some undesirable properties of this location. As the library and other university buildings are close, there were other WLANs present requiring the careful selection of the radio channel (a problem also described in [18]). Another issue were moving cars possibly leading to frequent changes in the topology even though the nodes themselves did not move.

To determine the connectivity between the nodes at the start of an experiment, each node transmitted a number of beacons. While one node transmits its beacon at a time, all other nodes record the packet reception. The necessary exact coordination is difficult in a distributed ad-hoc network but can be achieved with a *domino effect*. Prior to the experiment a route is determined that includes all nodes of the network. The nodes use the sequence imposed by this route to coordinate their beaconing. The first node starts with its beaconing. Its successor will take over once this node has finished. This is repeated until the last node has transmitted its beacons. The domino approach worked well for our small networks, however it can be foreseen that a more sophisticated mechanism is required for larger and more dynamic networks.

Based on the experiences gathered during the FleetNet project, we were aware of the problem of having to reimplement the algorithms when switching between simulations and experiments. Therefore, we implemented flooding in click [12]. With the help of nsclick [16], the same code used for the experiment can be run in ns-2 [17]. We used the integration for two purposes: the first is to debug and gain first experiences with the implementation and the setup of the experiment in a controlled simulator environment. When experiments are conducted without proper tests, this can waste a huge amount of manpower: a bug in the configuration of one of our in-building tests rendered a whole series of test runs useless. A miscalculation of the pause time between sending packets resulted in intermixed flooding attempts. A simulation prior to the testing could have revealed this problem without incurring significant overhead. The second reason for an integrated simulation and experimentation approach is to use the data gathered during an experiment to im-

prove simulations with information about real-world radio characteristics. To model the real setup as exactly as possible, also the positions of the nodes in the real experiment must be known. We used GPS for this purpose and encountered the well known problems of position jitter and dependence on clear view to the sky.

During our experiments it often happened that either a link, the used software or a whole node failed. Every time this happened, we had to check each node manually. As a provisional solution, we implemented a simple in-band one-hop status check. Each node wrote its current status to a file accessible via HTTP. To control the nodes' status, it was sufficient to walk around and use a perl script to retrieve the status file from each node. Obviously, this approach has several limitations: it requires to walk in the radio range of each node to be checked, the transmissions "contaminate" the experimental data, and the correction of an error still requires physical access to the affected node. Because of these experiences we believe that node monitoring will be a very valuable element of a good testbed.

Another issue appeared during the postprocessing of the data from the experiments and the simulations. As the output format of the simulator (ns-2) differs from the trace format of the experiment (tcpdump), each tool for the analysis of the results had to be implemented twice. Furthermore there exists currently no good solution for commentation and documentation of the raw data such that special events during the experiments can later on be remembered and reconstructed.

One key question occurred during the postprocessing: how good was the performance of the simple flooding strategy? Of course there are absolute values on the number of transmitted packets and the rate of received messages. However, this does not provide any clue on how good our approach worked under the given circumstances in comparison to an (illegal) optimal solution. Thus benchmarks would be a great help in determining how well a given solution performed.

4 Conclusions and Outlook

We believe that a MANET testbed should be open source, not restricted to special hardware, customizable, and not bound to any specific location. It must support reproducible, comprehensive, and correct experiments. While there are promising individual contributions towards this goal, currently no approach fully satisfies these demands. For a multitude of reasons the best way to implement such a testbed seems to be a joint effort of the MANET community. The most important one may be a broad acceptance by the community.

A first step towards this goal should be a specification of the functionality that the testbed must provide. We expect that this will be a controversial discussion and hope that this paper may contribute to this effort. Once the specification has become stable, the key factor to a successfull community effort will be the design of a highly modular system where each research group can contribute individual parts. We expect that the specification of scenarios, the scripting for running experiments and the format of the raw experiment data will require immediate attention and provide the glue for the connection of the individual contributions. In order to stimulate these first steps we have set up a wiki at *http://wikicn.cs.uni-duesseldorf.de*.

References

[1] D. A. Beyer. Accomplishments of the DARPA SURAN program. In *Proceedings of MILCOM'90*. IEEE, Sept 1990.

[2] C. T. Calafate, R. G. Garcia, and P. Manzoni. Optimizing the implementation of a MANET routing protocol in a heterogeneous environment. In *The Eighth IEEE Symposium on Computers and Communications*, June 2003.

[3] Emulab - mobile wireless networking. http://-www.emulab.net/tutorial/mobilewireless.php3.

[4] A. Farago and V. R. Syrotiuk. Merit: A unified framework for routing protocol assessment in mobile ad hoc networks. In *Proceedings of MobiCom'01*, pages 53–60. ACM Press, 2001.

[5] J. Garcia-Luna-Aceves. Wireless internet gateways (WINGs) for the internet. Technical report, University of California, Santa Cruz, 2001.

[6] gpsd: a GPS service daemon. http://gpsd.-berlios.de/.

[7] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer. Position-aware ad hoc wireless networks for inter-vehicle communications: The FleetNet project. In *Proceedings of MobiHoc'01*, Long Beach, California, October 2001.

[8] S. Jadhav, T. Brown, S. Doshi, D. Henkel, and R. Thekkekunnel. Lessons learned constructing a wireless ad hoc network test bed. 1st Workshop on Wireless Network Measurements (WINMee 2005), April 2005.

[9] J. Jubin and J. D. Turnow. The DARPA packet radio network protocols. In *Proceedings of the IEEE*, volume 75, pages 21–32, January 1987.

[10] G. Judd and P. Steenkiste. Repeatable and realistic wireless experimentation through physical emulation. *ACM SIGCOMM Computer Communication Review (CCR)*, 34(1):63–68, 2004.

[11] V. Kawadia, Y. Zhang, and B. Gupta. System services for ad-hoc routing: Architecture, implementation and experiences. In *Proceedings of MobiSys'03*, San Francisco, California, May 2003.

[12] E. Kohler, R. Morris, B. Chen, and J. Jannotti. The Click Modular Router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[13] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of MSWiM'04*, pages 78–82, October 2004.

[14] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin. A large-scale testbed for reproducible Ad Hoc protocol evaluations. In *Proceedings of WCNC'02*, pages 337–343, Orlando, FL, March 2002.

[15] D. A. Maltz, J. Broch, and D. B. Johnson. Experiences designing and building a multi-hop wireless ad hoc network testbed. Technical Report CMU-CS-99-116, School of Computer Science, Carnegie Mellon University, 1999.

[16] M. Neufeld, A. Jain, and D. Grunwald. Nsclick: Bridging Network Simulation and Deployment. In *Proceedings of MSWiM'02*, pages 74–81, Atlanta, Georgia, September 2002.

[17] The ns-2 network simulator. http://www.isi.edu/-nsnam/ns/.

[18] K. Ramachandran, K. Almeroth, and E. Belding-Royer. A novel framework for the management of large-scale wireless network testbeds. 1st Workshop on Wireless Network Measurements (WINMee 2005), April 2005.

[19] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *Proceedings of WCNC'05*, New Orleans, LA, March 2005. IEEE. (to appear).

[20] A. Tarp. Experimental evaluation of flooding in ad-hoc networks. Bachelor thesis, 2005. Department of Computer Science, University of Düsseldorf.

[21] tcpdump: a tool for network monitoring, protocol debugging and data acquisition. http://www.-tcpdump.org.

[22] Y. Zhang and W. Li. An integrated environment for testing mobile ad-hoc networks. In *Proceedings of MobiHoc'02*, pages 104–111. ACM Press, 2002.

[23] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of MobiSys'04*, pages 125–138. ACM Press, May 2004.

Experiences Deploying an Ad-hoc Network in an Urban Environment

Peter Barron, Stefan Weber, Siobhán Clarke, and Vinny Cahill
Distributed Systems Group,
Department of Computer Science,
Trinity College,
Dublin 2, Ireland.
email: {Peter.Barron, Stefan.Weber, Siobhan.Clarke, Vinny.Cahill}@cs.tcd.ie

Abstract

Studies of mobile ad-hoc networks are in the most part restricted to simulations and theory. They have, to this point, rarely ventured into the real world on a large enough scale to make significant statements about their behavior or performance. The lack of evaluations is largely due to the practicable implications of deploying and evaluating such networks in a real environment. In order to analyse the problems we have built a Wireless Ad-hoc Network for Dublin (WAND). The network provides a large-scale testbed for applications and protocols for mobile ad-hoc networks. It offers the opportunity to explore the behavior and performance of a variety of routing protocols in a real-life environment and an ideal platform for investigating the use of mobile applications in an urban environment. In this paper we present WAND and the experiences gained in building such a network.

1. Introduction

Much of the research in ad-hoc networks has been restricted to the evaluation of solutions through simulation. The simulators used typically aim to represent the different software and hardware components within the system as well as the physical environment in which they operate. While this provides a useful method of validation the simplified assumptions made of the physical environment limit the scope of what can be achieve from them [4, 8]. For instance, the phenomena of gray-zones were only discovered through experiments conducted real world environment. It is therefore necessary to complemented simulation studies with real-world experiments to identify phenomena that would otherwise go unnoticed within a simulator. For these reasons, we have built WAND - the Wireless Ad-hoc Network for Dublin - a large-scale testbed for ad-hoc network protocols and applications.

The network covers the centre of Dublin along a 2km route from Trinity College Dublin to Christ Church Cathedral. The area is seeded with a number embedded devices with wireless connectivity. These devices have been custom-built to meet the needs of the WAND network. They are housed in 3x3x6inch containers that accommodate a stack of PC/104 boards, IEEE 802.11b PCMCIA cards, and two patch antennae. The embedded devices are hosted on traffic lights and cameras along the route to provide a minimum level of connectivity. The embedded devices form a sparse population of wireless network nodes. This sparse coverage is constantly available and the embedded devices can be configured to create a variety of network models. The testbed can be further populated through the introduction of mobile nodes such as laptops, PDAs, and other mobile devices with wireless connectivity. The network provides researchers with a flexible platform for developing and investigating different protocols and applications for ad-hoc networks.

In designing the WAND network, the goal has been to provide a platform that would allow basic research on the behavior and performance of ad-hoc routing protocols in a real-life environment. The choice of an urban setting, while also providing a fertile en-

vironment for mobile applications, exposes the network to the problems and challenges of operating in an environment that is inherently unpredictable. The changes in weather, movement of people, and that of cars, trucks, and buses, while the presence of buildings all have an effect. Rather than avoiding these issues we aim to include and measure their effect on the behaviour of the network and its components. The WAND testbed also offers a unique opportunity to explore the different aspects of building applications for wireless ad-hoc networks in an urban setting. The goal has also been to use the WAND network to build integrated traffic systems, pervasive computing applications, and location-based services, and as such, it provides an excellent testbed for investigating these areas of research.

In this paper we discuss the technology employed in the test-bed and the experiences gained in deploying the WAND network. We then present some initial results of the throughput over a series of nodes using AODV [14].

2. Description of WAND

The section provides details of the WAND testbed. Describing the environment that it is operates in, the hardware that is used to form the initial sparse population of fixed nodes for the network, and the software that is installed on each of these nodes.

The testbed is locate at the heart of the shopping and business districts of Dublin along a 2km route. It is seeded with a number of custom-build wireless-enabled embedded PCs. The nodes are distributed along the route in the form of a line at a distance of about 200m apart. Each pair of neighbouring nodes is installed within line of sight of each other. There are currently eleven of these nodes located along the route. The nodes are mounted on traffic lights and camera positions at a height of about 3m. Figure 1 provides an overview of the area covered and the distribution of the nodes. The *GK* node is used as gateway from the WAND network to the college network.

The streets the network is deployed on are approximately 16-22 m wide and are lined by buildings of 4-6 stores high. The majority of these buildings have thick stone-walls that block the propagation of radio signals onto the neighbouring streets running in parallel.

Figure 2. Hardware used to WAND node.

The streets in the area are open to public traffic which run on either one- or two-way streets. The vehicular traffic on these streets consists of a mixture of cars and buses; articulated trucks do not enter this area of the city centre. Most of the vehicles have a height between 1.5m and 2.5m except for double-decker buses which have a height of 4.40m.

2.1. Hardware

The hardware for the nodes that form the initial sparse population for the WAND network are custom-build embedded PCs. Each node consists of a stack of PC104 boards [13, 12], a number of 2.4GHz PCMCIA wireless cards, an enclosure, and a set of antennae. The stack of PC104 boards comprises of a motherboard and two PCMCIA PCI-boards (see Figure 2).

These boards are mounted inside an enclosure together with a 30GB hard-drive. The motherboard employed in the system is a 400-700 MHz EEPD C3VE board featuring a Pentium-class processor, 128MB of memory, a video and an Ethernet adapter as well as an IDE controller. Each of the PCMCIA boards have two PCMCIA slots. These slots are filled by three CISCO Aironet 350 cards which provided 802.11b connectivity. The cards have two sockets that connect to external antennae instead of an internal antennae which is more commonly used. The sockets are connected via 10cm cables to RP-TNC connectors that are mounted on top of the enclosure. The setup allows a variety of antennae to be used by the nodes. At the time of writing, each of the fixed nodes is equipped with two patch-panel antennae with a gain of 8.5 dBi. The radiation pattern of these antennae is in the shape of a

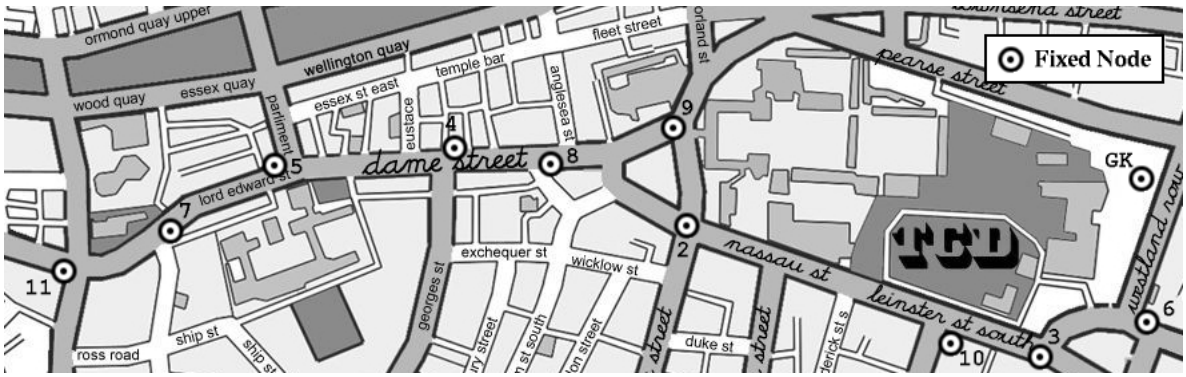


Figure 1. Map of the area covered by WAND.

hemisphere in front of the antennae. The patch-panel antennae of neighbouring nodes are directed towards each other.

2.2. Software

The platform provided by the nodes is similar to what your might find on a Pentium based PC with a number of extra wireless network cards connected. At present, the nodes have been installed with Red-Hat Linux 9. The installation includes packages such as web-server, a JDK installation (version 1.4.2), a secure-shell daemon, plus the majority of the other packages that typically come with this distribution.

The kernel, that by default comes with the Red-Hat Linux 9 installation, has been replaced with the kernel from the RedHat Linux 7.3 distribution. This is due to the RedHat 9, kernel version 2.4.20, using PCI-hotplug which conflicts with the current configuration of the PC104 stack. The kernel that comes with the distribution of RedHat Linux 7.3, kernel version 2.4.18-3 was developed prior to the introduction of PCI-hotplug support in the kernel and operates without problems on the PC104 stack.

Currently, the OLSR [7], and AODV [14] routing protocols have been installed on the nodes. The AODV implementation - Kernel AODV (version 2.1) - is from the National Institute of Standards and Technology and the OLSR implementation is from [3]. It is possible to install other protocols though at this stage we have not done so.

2.3. Configuration scenarios

The aim has been to provide a flexible platform that can support various network topologies and installation of different routing protocols. It is possible to configure the sparse network of nodes, that make up the WAND network, to provide a backbone and serve as regular access points to mobile nodes. Alternatively, they can also be configured to use the wireless cards and antennae they have to resemble a network of ad-hoc nodes. Mobile nodes can then be connected to form larger ad hoc networks.

The fixed nodes also provide a excellent platform for running different pervasive computing, mobile applications, or integrated traffic systems scenarios. The embedded devices used are ideally placed in the middle of the shopping, tourist, and business districts of Dublin and have enough power to support applications that require quality-of-service support or extra processing power. The fixed installation of nodes also ensures that a permanent presence can be maintained for example for location-aware services.

3. Description of Experiences

This section describes some the experiences gained during the installation and preliminary evaluation and use of the testbed. The issues listed are broken into two categories. Those that occurred during the initial installation and those that subsequently arose from the day-to-day use of the network as a testbed for ad-hoc routing protocols, and for application development and testing.

3.1. Installation issues

The installation of the testbed involved an initial design phase followed by the deployment of the nodes between Trinity College Dublin and Christ Church Cathedral. The choice of hardware and software used was based on an initial set of requirements for providing a flexible testbed for basic research into protocols and applications is a wireless ad-hoc network. The deployment of the nodes was achieved in collaboration with the local authority who provided access to the traffic lights and camera positions along the length of the route. The experiences gained from building WAND have shown to us many of the difficulties facing those looking to develop these types of networks. In this we wish to highlight a number of them.

Hardware is difficult. During the initial design of the testbed a number of hardware configurations were considered. One of the major factors that influenced the design was the availability of off-the-shelf hardware for IEEE 802.11a/b/g network cards. Most of the wireless network cards either available in either PCMCIA or cardbus format. Thus the hardware of a fixed node had to support the PCMCIA format.

A number of community wireless projects use PCBs with one PCMCIA slot. One of the intend use for the testbed is the development and evaluation of multimedia applications and quality-of-service (QoS) protocols. The specifications for these uses required more processing power and storage capacity than the available PCB could provide.

We finally choose the current hardware configuration because of its compatibility to desktop PCs, availability of off-the-shelf components and the ability to configure it to suit the required number of network cards. However, the lack of previous experience with this hardware platform and the lack of maturity of the hardware at the time lead to a number of problems: Motherboards that comply with the PC104 standard and are based on Intel 80386 processors are relatively mature and stable. High-end motherboards that comply with the PC104+ standard and feature Pentium-level processors are more complex. The PCI-bus that was introduced with the PC104+standard requires components in a PC104+ stack to accurately synchronize the timing of signals on the bus. This requirement makes the use of PC104+ stacks more complex and difficult compared to PC104 stacks.

Positioning of nodes. When deploying the network between Trinity College Dublin and Christ Church Cathedral it was necessary to find the best positions along the route for the nodes. However, we found that there where a number of issues that restricted where we positioned the nodes. First, was the actual range that could be obtained from the nodes and the patch antennae being used. It was found that the distance could vary considerably depending on the buildings, density of vegetation, road congestion, and interference from existing networks in the area. However, the biggest influence was the actual route taken by the network in combination with height of the buildings in the surrounding area. In many cases the line of sight between nodes was lost as the network weaved it way through the city centre. The combination of factors significantly decrease what could normally be achieved when line of sight could be maintained. Sources of power for the nodes also provided to be an issue that restricted where we could place the nodes. The only power sources available to us were the traffic lights and some of the camera positions locate along the route. This provided us with 17 possible locations for 11 WAND nodes along a 2km route. However, we found a number of issues that limited the number of possible positions/locations for the nodes. The use of other sources such as telephone kiosks, apartments, shops would have enabled a better distribution of the nodes and provided better connectivity; unfortunately these locations were not available to us at the time of the installation.

Interference from other sources. The location chosen for the network, while providing an excellent testbed for mobile and pervasive computing research, is in the middle Dublin City Centre at the heart of the business and shopping districts. A wireless survey, using kismet, showed a large number of networks, approximately 35, already existing in the area. The majority were infrastructural based but there were also a couple ad-hoc networks. To annoyed interference with these networks it was necessary to choose a channel not being use by these networks. Unfortunately, due the size of the WAND network the survey showed

that the majority of channels were already in use. It was therefore necessary to pick a frequency that would cause the lease amount of interference with the existing networks to ensure the WAND network could operate successfully.

Because of the unlicensed nature of the 2.4GHz band this will increasingly become a problem for deploying these types of networks in urban environments. There is also number of other sources of interference that can cause problems and which are not so easily detected. An example for one such source is the coordination of traffic cameras that are installed at various points along the route. These cameras are controlled generally through a wired connection but sometimes the last 100m to a camera pole is based on wireless communication in the 2.4GHz.

Height of installation The height of the installation of the fixed nodes is a compromise between security of the hardware installation, the quality of the RF signal and the accessibility of the hardware for maintenance. One of the initial concerns regarding the height of the installation was the clearance of the Fresnel zone. A rule of thumb determines that 60% of the 1st Fresnel zone should be free of obstacles. This meant that the nodes needed to be installed at least 1.5m above every obstacle given that the fixed nodes were to be installed every 200m along the route. An installation height of 4m would have been ideal to locate the direct line of sight in sufficient space over the heads of pedestrians and to protect the nodes against possible vandalism. However, this height was deemed to be inaccessible for maintenance purposes. It was predicted that due to the exploratory nature of the installation the nodes needed to be accessible without great difficulty and so a compromise was made to install the nodes at a height of 3m. This height allows access to the nodes by ladders and gives enough clearance of the Fresnel zone to provide a good signal. One of the main obstacles due to the height of the installation is the interference of buses. Double-decker buses have a height of 4.20m and tend to block the line of sight between two fixed nodes completely.

3.2. Maintenance issues

A number of projects and experiments were run following the deployment of the testbed. The experiences gained from running them on the WAND network are described within the following sections.

Configuration of services The wide distribution of the WAND nodes along with the dynamic state of the network has lead to difficulties in maintaining the configuration of the network in a sustainable way. Currently, access to the nodes is achieved either from a gateway node located at one end of the network, or via other wireless devices - laptops, PDAs - within the WAND network itself. Maintenance and the continuing configuration of the WAND nodes can only be accomplished through either of these routes. However, the preferred route of administration has to be through the gateway node as updating nodes from mobile devices within the network itself is impracticable over a sustained period. The problem we have found is that all the nodes are not always accessible from the gateway node due to a variety issues. Therefore, configuring or updating services or the network configuration is difficult to achieve throughout the network without it failing. So far, we have not been able to find a satisfactory mechanism that would allow updates to be propagated through the network and be applied unilaterally.

Development of applications A number of projects have been based on the testbed. Cocoa [2], Visualisation, Multimedia between mobile devices using OLSR. The use of the WAND network has proven to be a valuable resource in understanding how to build these types of applications. The sparse network of nodes provides a stable level of connectivity for these projects that could not otherwise be achieved without a greater population of mobile nodes within the environment. However, the deployment of these experiments and the retrieval of results from the nodes has proven to be challenging. Again, the accessibility to all the nodes from one location has proven to be the main problem. To provide a usable testbed it is necessary to be able to coordinate and configure the network from one set of experiments to another. To be able to

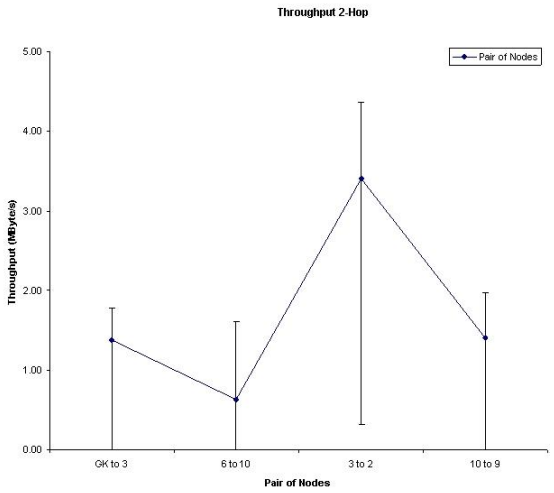


Figure 3. Throughput over two hops.

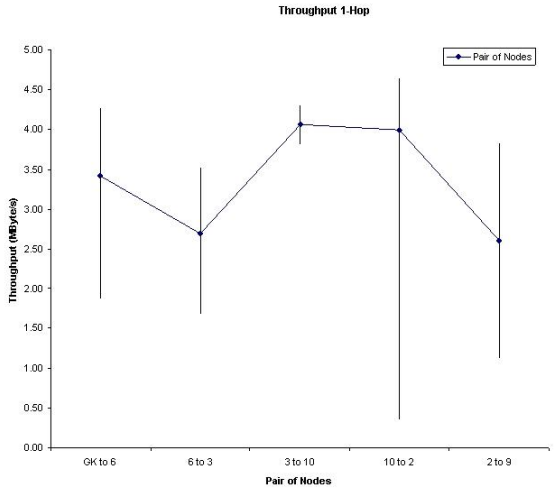


Figure 4. Throughput between individual pairs of nodes.

deploy, configure, and retrieve the results in an effective manner. For the WAND network this has proven to difficult to achieve due to the unpredictable nature of the network itself.

4. Evaluation

This section describes a number of preliminary experiments used to examine the throughput of the WAND network using the AODV implementation from the National Institute of Standards and Technology over a multi-hop network.

4.1. Throughput

The graph in figure 4 shows the throughput between individual pairs of nodes. In order to understand this output correctly it is necessary to consider at the location of the individual nodes. The first pair of nodes has direct line of sight which results in good throughput of around 3.5 MB/s. However, the nodes are installed on different sides of the streets and the traffic on the street causes interference and leads to a variation in the throughput. The line of sight between the next pair of nodes is interrupted by a bend in the road. This causes a slight drop in the throughput between the two nodes; additionally the traffic on this street causes the throughput to fluctuate as well. The third pair of nodes is located on the same side of a straight street. This setup with no interference through traffic or obstacles in the line of sight results in high throughput with little variance. The next set of nodes is located on the same road, slightly further apart. The throughput between these two nodes is equally good to the previous pair despite the increased distance; however, a junction before node 3 where a large number of buses turn off from the main road leads to large variation in the throughput. The line of sight of last pair of nodes is again disrupted by a bend in the road and the throughput shows a similar pattern to the throughput of the second pair.

The second graph (figure 3) shows the throughput over a 2-hop chain. The first chain involves a pair of nodes with a clear line of sight and a pair of nodes that a separated by a bend in the road. This results in relatively low throughput with great variance. The next chain has the pair of nodes separated by a bend as first pair followed by a pair that has a clear line of sight. This results in less throughput than in the first case with similar variance. The third chain has two pairs of nodes with a clear line of sight; however one of the pairs is separated by the junction with a high amount of bus traffic that was mentioned earlier. This setup results in high throughput but with great variance due to the bus traffic. The fourth chain is similar to the first chain in that the first pair in this chain has a clear

line of sight - sometimes interrupted by traffic and the second pair is separated by a bend in the road. This again results in low throughput with great variance.

From these two experiments we can see first indications of the influence of the traffic on the roads on the throughput. Also, these measurements show the effect of the obstructions in the line of sight and its influence on the throughput in 2-hop chains.

The influence of the number of environmental factors such as weather, time of day, etc has not yet been investigated and requires further measurements.

4.2. Related Work

The work related to the experience reported here can be categorized into 3 general areas: reports on community networks, reports on testbeds targeting research in mobile ad hoc network (MANET) protocols and reports on testbeds targeting the investigation of sensor networks.

Community networks such as MIT's Roofnet [1] employ similar hardware to the one used for WAND nodes. Roofnet provides an experimental 802.11b/g mesh network which offers broadband Internet access to participants. The nodes in this network are fixed in a place and consist of small PCs with a wireless and a wired interface. The wireless interface is generally connected to a roof-mounted omni-directional antenna. The nodes provide routing between mobile nodes and the internet. Roofnet provides a platform for experiments to understand the nature of large-scale wireless networks. The research focus of this network is the study of link-level characteristics of 802.11 and the development of high-throughput routes.

A number of research projects have proposed setups of testbeds in order to evaluate protocols for MANETs [5]. Maltz et al [10, 11] report their experiences during an experimental evaluation of Dynamic Source Routing (DSR) protocol under the influence of a high-rate of topology changes. The testbed for this evaluation consisted of 8 nodes that comprised IBM Thinkpads 560X notebooks, a 900 MHz WaveLAN-I radio with a 6db omni-directional antenna and a Trimble 7400 GPS receiver. The nodes were mounted in cars that were driven in a set course at an urban road. By evaluating the routing protocol in a testbed the researchers developed a number of new ideas for the improvement of the protocol.

Tschudin et al [16] developed the Ad hoc Protocol Evaluation (APE) testbed. This testbed is based on a software package in form of a Linux distribution. Laptops can be booted from a CD using this package. Tschudin et al report on experiments with up to 37 nodes and 3 routing protocols [9]. The evaluation of their implementation of the Ad hoc On-Demand Distance Vector (AODV) routing protocol in a real-world environment led to the discovery of gray-zones. In these gray-zones a routing protocol would report a valid route to a destination but almost no data would be delivered to the destination. This phenomenon was not encountered in simulations of the same routing protocol and provides one of the many motivations for evaluations of protocols in real-world scenarios.

The third area that is related to the experience reported here is the area of sensor networks. Ganesan et al [6] describe the experience they gained while evaluating a sensor network of 185 nodes in an open park area. The nodes are based on a 4MHz ATMEL ATmega with a 900MHz low power radio. They found that while current RF-propagation models can be used to model certain behaviour such as that of sparse wireless networks; they need to be extended for dense deployments such as dense sensor networks. Ritter et al [15] describe a similar network based on a combination of a microcontroller and a number of wireless interfaces such as 433MHz RF and Bluetooth modules. This envisioned application domains for this network are ad hoc gaming and home automation.

5. Conclusion

In this paper we have presented a testbed for the development and evaluation of ad hoc network protocols and applications within an urban environment. The experiences that we have gathered through the development and deployment of this testbed are various. It is extremely difficult to design a complex and flexible testbed that delivers reproduceable yet realistic results.

However, we believe that no simulation is able to reflect the inherent uncertainties and erratic nature of real-world environments. The behavior predicted by simulation may vary dramatically from that observed in real networks such as WAND. Thus the measurements done in such a testbed will complement our un-

derstanding of the behavior and performance of protocols in realistic environments and the appropriate design and implementation of applications for such areas as pervasive computing.

6. Acknowledgments

The work described in this paper has been partly supported by the Irish Higher Education Authority under the Carmen collaboration between Media Lab Europe and Trinity College. The authors are also grateful to Dublin City Council for supporting the installation of the WAND nodes and for Jade Mastersons for the use of the map of Dublin.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of the ACM SIGCOMM'04 Conference*, Portland, Oregon, August 2004.

[2] P. Barron and V. Cahill. Using stigmergy to coordinate pervasive computing environments. In *Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*, pages 62–71, December 2004.

[3] C. Calafate, R. Garcia, and P. Manzoni. Optimizing the implementation of a manet routing protocol in a heterogeneous environment. In *Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC'2003)*, Kemer - Antalya, Turkey, 2003.

[4] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *Proceedings of the Workshop on Principles of Mobile Computing (POMC'02)*, pages 38–43. ACM, Oct. 2002.

[5] P. De, A. Raniwala, S. Sharma, and T. cker Chiueh. Mint: A miniatuirized network testbed for mobile wireless research. In *Proceedings of IEEE Infocom 2005*, Miami, Florida, USA, March 2005. IEEE.

[6] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report UCLA/CSD-TR 02-0013, University of California Los Angeles, 2002.

[7] P. Jacquet, P. Muehlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *IEEE International Multi Topic Conference INMIC 2001*, Lahore, Pakistan, 2001.

[8] D. Kotz, C. Newport, R. S. Gray, J. L. andYougu Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proceedings of the ACM/IEEE International Symposium on Modeling,Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 78–82, October 2004.

[9] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordstroem, and C. Tschudin. A large-scale testbed for reproducible ad hoc protocol evaluations. In *Proceedings of 3rd annual IEEE Wireless Communications and Networking Conference (WCNC 2002)*, pages 412–418, Orlando, Florida, USA, March 2002. IEEE.

[10] D. Maltz, J. Broch, and D. Johnson. Experiences designing and building a multi-hop wireless ad hoc network testbed. Technical Report CMU-CS-99-116, Carnegie Mellon University, March 1999.

[11] D. Maltz, J. Broch, and D. Johnson. Lessons from a full-scale multi-hop wireless ad hoc network testbed. *IEEE Personal Communications*, 8(1):8–15, February 2001.

[12] C. PC/104 Embedded Consortium, San Jose. Pc/104-plus specification version 1.2, August 2001.

[13] C. PC/104 Embedded Consortium, San Jose. Pc/104 specification version 2.5, November 2003.

[14] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[15] H. Ritter, T. Voigt, M. Tian, and J. Schiller. A highly flexible testbed for studies of ad-hoc network behaviour. In *Proceedings of International Workshop on Wireless Local Networks (WLN2003)*, Bonn/Koenigswinter, Germany, October 2003.

[16] C. Tschudin, P. Gunningberg, H. Lundgren, and E. Nordstroem. Lessons from experimental manet research. *Elsevier Ad Hoc Networks Journal*, pages 221–233, March 2005.

MeshDV: A Distance Vector mobility-tolerant routing protocol for Wireless Mesh Networks

Luigi Iannone Serge Fdida
LIP6/CNRS – Université Paris VI – Paris – France
{luigi.iannone,serge.fdida}@lip6.fr

***Abstract*— In this paper we propose an overview of MeshDV, a routing protocol for wireless mesh networks. MeshDV combines proactive route computation for routers and on-demand path setup for clients. This design choice eases the management of client mobility and reduces routing table size. Proactive route computation is performed using a Distance Vector approach. On-demand path setup is obtained through new mechanisms that take advantage of particular characteristics of mesh networks. Here we focus on these new mechanisms, giving details about algorithms, packet format, and packet exchange procedures. We also present the main components of the MeshDV prototype currently under development at the LIP6 laboratory of the University of Paris VI.**

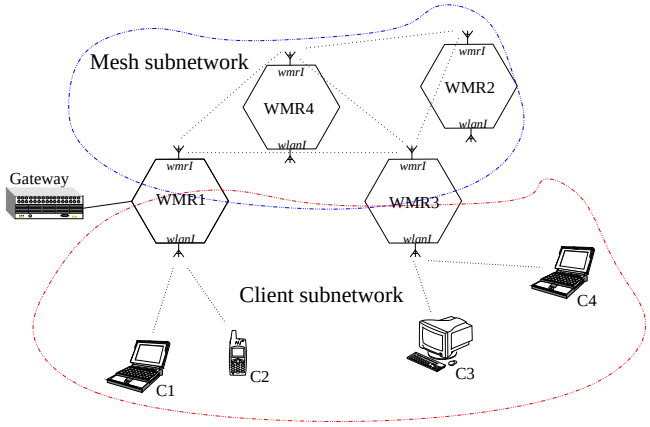


Fig. 1. An example of Wireless Mesh Network deployment.

I. Introduction

The natural evolution of wireless networks enables new ways for mobile users to get connected to each other. In this context, Wireless Mesh Networks (WMNs) are a promising emerging technology. In WMNs, nodes are composed of wireless mesh routers (WMR) and mesh clients.[1] The architecture of WMNs is two-tier, composed of a subnetwork for clients and a subnetwork for mesh routers. The client subnetwork has the purpose of offering wireless access to any client allowed to access the network. The mesh routers subnetwork is the backbone used to route packets between clients associated with different WMRs or between clients and gateways. The gateways between the client and the mesh subnetworks are the wireless mesh routers. Figure 1 shows an example of this type of mesh network.

Routing in such a complex and dynamic structure exhibits different, still unsolved, technical challenges. Despite the large amount of research performed, few works have been actually implemented. Typical proposals simply apply to WMNs existing solutions, proposed in other contexts. For instance, LocustWorld [3] uses a standard linux-based AODV implementation, and the Roofnet Project at MIT [4] uses a network address translator (NAT) on each WMR. The management of client mobility can be done either through mobile IP or by embedding routing features in the clients. Mobile IP has drawbacks, like triangular routing, which makes it not suitable for WMNs. The second solution works with a flat, ad hoc architecture and limits the access solely to clients able to support routing.

MeshDV, our proposed architecture, is designed to take full advantage of WMN architecture. MeshDV mixes proactive route computation for routers and on-demand path setup for clients. This design eases the management of client mobility and reduces routing table size. Proactive route computation is performed using a Distance Vector (DV) approach. On-demand path setup is performed by new extensions introduced in the routing protocol in order to take advantage of the architecture of WMNs.

In MeshDV, the backbone becomes totally transparent to the clients, which do not need to embed any new feature. This means that if two clients associated to different WMRs wish to communicate, the set of WMRs will forward the traffic at the IP level and not at the link level. To obtain such a behavior, WMRs have to be more than a simple forwarder and more than a simple router. In particular, on the client subnetwork interface,

[1]For simplicity of presentation we will call mesh clients merely clients.

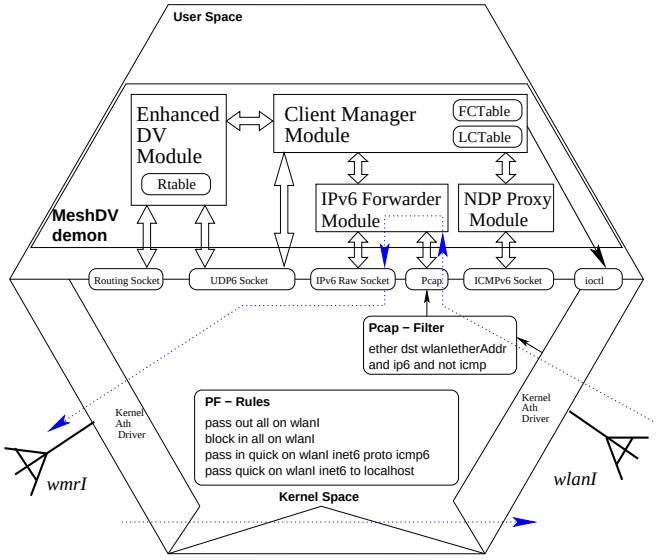


Fig. 2. WMR functional architecture.

the WMR acts in such a way to let local clients think that remote clients are in the local WLAN. Then, it is up to the WMRs to find out to which WMR the client is associated and route the packets accordingly.

The University of Paris VI is planning to cover the whole university campus with a large IPv6-based wireless mesh network. `MeshDV` represents the prototyping and research work developed at LIP6, for the deployment of a mesh routing protocol.

The remainder of the paper is organized as follows. In section II, the general architecture of `MeshDV` and the communication setup are presented. In sections III to VI, the main modules of `MeshDV` are described. Details about the deployed testbed, and some preliminary results are given in section VII.

II. `MeshDV` Overview

The fundamental blocks of WMNs are WMRs, whose functional architecture is depicted in Figure 2. Each WMR has at least two wireless interfaces, one for the client subnetwork and one for the mesh subnetwork. We call the first interface *wlanI*, since it implements a WLAN offering access to clients, and the second one *wmrI*, since it permits communications between WMRs.

Next we describe the architecture of `MeshDV` and then we show how communication are set up.

A. `MeshDV` *Architecture*

The general architecture of `MeshDV` is also presented in Figure 2; dotted lines represent the data flows between the two interfaces inside the WMR. Each module of the presented architecture has a specific meaning and task.

The basic task of the Enhanced DV module is to maintain proactively a route toward any other WMR present in the backbone subnetwork. This Enhanced DV module is an IPv6 [5] implementation of DSDV [1], based on existing IPv4 code [2]. Some enhancements have been introduced in order to cooperate with the Client Manager module. This module is the only one that accesses the kernel routing table.

The Client Manager module is responsible for discovering which client is associated to which WMR in an on-demand fashion. When a local client wishes to communicate to a remote client (associated to another WMR), it sends a multicast request in order to discover where the client is.[2] The result of this query is stored in the Foreign Client Table (`FCTable`). The Client Manager module also manages the reply to requests sent by other WMRs. Furthermore, since `MeshDV` must be aware of the clients that are associated to the WMR, the Client Manager module monitors the set of clients associated to *wlanI* and stores them in the Local Client Table (`LCTable`).

The NDP-Proxy module provides the WMR with the ability to act like a Neighbor Discovery Protocol proxy. This allows to transparently forward packets toward the WMR the destination client is associated with. In this way, no particular mechanisms are necessary on the client side, in order to communicate with remote clients. The NDP-proxy module will correctly answer to the ICMPv6 request sent by the local clients.

The IPv6 Forwarder module encapsulates all IPv6 packets in another IPv6 packet, in order to ship packets toward the destination client. This solution, though introducing a certain amount of overhead, avoids keeping state in the WMRs along the path between clients. Indeed, only the WMRs at the edges of the path must be aware that the two clients are communicating. Another significant advantage is the robustness to mobility. If a client changes the WMR to which it is associated, only the WMRs at the edges of the path must update the information (3 WMRs in total). WMRs along the path do not need to make any update, while continuing to relay packets.

In the next section we provide an example of how communications are set up in a Wireless Mesh Network.

B. *Communication Setup*

Traffic flowing in a mesh network can be classified into three types: a) traffic between two clients associated

[2]Recall that in IPv6 broadcast addresses do not exist anymore, multicast is used to send packets to multiple receivers.

with the same WMR, b) traffic between two clients associated with different WMRs, and c) traffic between a client and a gateway. All types of traffic are detailed below.

Clients associated with the same WMR. In this case, clients do not need any particular attention, since the *wlanI* interface automatically bridges the traffic between them. This is an embedded feature of the HostAP mode of wireless card drivers.

Clients associated with different WMRs. Let us take the scenario of Figure 1 and suppose that client C_1 wants to send a packet to client C_4. The following steps show how the communication between C_1 and C_4 is set up, according to `MeshDV` rules:

1: C_1 sends an ICMPv6 *Neighbor Solicitation* packet for C_4.
2: The NDP-Proxy module of WMR_1 receives the packet from C_1 and sends a message to the Client Manager module, requesting to find out which WMR C_4 is associated with.
3: The Client Manager module of WMR_1 checks in the `LCTable` and `FCTable` tables if the address of C_4 is present. If it is not, it sends a multicast packet (named `MCREQ`) on its *wmrI* interface, in order to ask the other WMRs if someone knows C_4.
4: The Client Manager module of WMR_3 receives the multicast request and checks its own associated client list. It knows C_4, thus it sends a unicast reply (named `CRREP`). WMR_3 keeps track, in its local client table, that it exported the address of the client.
5: The Client Manager module of WMR_1 receives the unicast reply. It stores in its foreign table that C_4 is associated to WMR_3 and sends a message to the NDP-Proxy module to announce that it knows to which WMR the client is associated.
6: The NDP-Proxy module of WMR_1 sends to C_1 an ICMPv6 *Neighbor Advertisement* packet associating the IP address of C_4 to the Ethernet address of the *wlanI* interface.
7: C_1 receives the ICMPv6 packet and now sends the data packet for C_4 to WMR_1.
8: The data packet is captured by the Forwarder module of WMR_1, which encapsulates it in a packet destined to WMR_3 and sends it.
9: The netBSD kernel of WMR_3 receives the packet, decapsulates it and automatically forwards it on the local *wlanI* interface where C_4 is associated.
10: C_4 receives the data packet.

Figure 3 shows the temporal diagram for the above mentioned exchange, which we call client advertisement mechanism. Note that since the proposed mechanism is symmetric, if C_4 needs to send back a packet, the same mechanism applies.

Client-gateway traffic. Traffic from a client toward a gateway does not need any particular action, since the Enhanced DV Module performs also Gateway advertisement. Traffic from the gateway toward a client follows the same rules like traffic between clients associated to different WMRs.

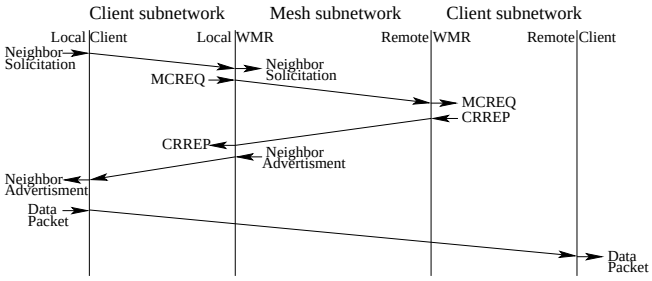


Fig. 3. Communication set up temporal diagram for clients associated to different WMRs.

III. Client Manager Module Data Structures and Packet Formats

In this section, we provide details about the data structures of the Client Manager module, namely the Local Client Table (`LCTable`) and the Foreign Client Table (`FCTable`) and outline how they are managed. Moreover, we present the set of packets necessary to perform client advertisement, listing the format and the purpose of each packet type.

A. Local Clients Table

The `MeshDV` routing demon is aware of the clients that are associated to the WMR on which it is running. For this purpose, it monitors the set of clients associated with *wlanI* and stores them in the Local Client Table (`LCTable`), which has the following format:

LCTable		
ClientIp	ClientMAC	ExternWMRs
`Associated ClientIp 1`	`MAC address Client 1`	`WMR_X Active`
`Associated ClientIp 2`	`MAC address Client 2`	`NULL`
`Associated ClientIp 3`	`MAC address Client 3`	`NULL`
.....	.....	.....
`Associated ClientIp N`	`MAC address Client N`	`NULL`

Each entry contains the IPv6 address of the associated client, its MAC address, and a pointer to a list. The list contains the IPv6 address and state information for all the WMRs that have asked for that client, *i.e.*, that have sent a client request packet. Otherwise the content of the field is `NULL`. State information is related to the timeout mechanism, which is detailed in [12]. In the above table, for example, the WMR has received from `WMR_X` a request for Client1 and nothing for the rest.

`MeshDV` updates the table of associated clients by accessing periodically the kernel, using `ioctl` calls. After an update, some new clients may have associated and some may have departed. For a newly associated client, its IPv6 and MAC addresses are added to the `LCTable` and the `ExternWMRs` field is initialized to `NULL`. For a departed client, if `ExternWMRs` is not `NULL`, for each WMRs in the list, a withdraw request (`CWIT`) is sent. Once all WMRs in the list have replied,

the entry of the client is deleted from the table. An LCTable update may also be triggered by the reception of client request packets. In this case, the IPv6 address of the request sender is added to the ExternWMRs list of the entry corresponding to the requested client.

B. Foreign Clients Table

When a client wants to communicate to a client associated elsewhere, there is the need to know which other WMR manages the destination client. MeshDV holds this information in the Foreign Client Table (FCTable), which has the following format:

FCTable	
ClientIp	ManagerWMRIP
Foreign ClientIp 1	Manager WMR IP of Client 1
Foreign ClientIp 2	Manager WMR IP of Client 2
Foreign ClientIp 3	Manager WMR IP of Client 3
.....	.....
Foreign ClientIp N	Manager WMR IP of Client N

Each entry contains the IPv6 address for the foreign client and the IPv6 address of the WMR with which the foreign client is associated. The latter field is a list that may contain several IPv6 addresses. Indeed, due to transient phenomena caused by mobility of the nodes, more than one WMR can send a CRREP packet to advertise itself as manager for the same client. Except for particular transient conditions, which, however, have limited lifetime, this field always contains only one WMR address. Transient conditions are described in [12].

MeshDV updates this table upon receiving or sending particular packets. When the WMR sends a MCREQ for a foreign client, a new entry is added to the table. The IPv6 address of the client is put in the ClientIP field and the ManagerWMRIP field is initialized to NULL. When the WMR receives a CRREP for the foreign client, the IPv6 address of the reply sender is added to the ManagerWMRIP list. When the WMR receives a CWIT for a client, the IPv6 address of the packet sender is deleted from the ManagerWMRIP list. If the list becomes empty, the whole entry is deleted from the table.

C. Client Advertisement Packets

Here we present the format of the packets used to perform client advertisement. For each packet we just give its purpose, format, and the content of each field of the packet. Their use is detailed in section V.

1) MCREQ: The Multicast Client REQuest (MCREQ) packet is issued whenever a WMR needs to know which WMR a client is associated with. As the name suggests, this packet is always multicast. The format is presented in Figure 4, and the content is:

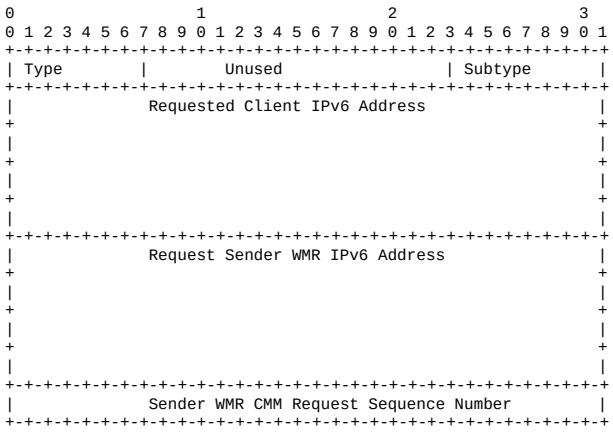


Fig. 4. MCREQ/UCREQ Packet format.

Type: Set to 2, Client Advertisement Mechanism.
Unused: Sent as 0; ignored on reception.
Subtype: Set to 1, Multicast Client REQuest (MCREQ).
Requested Client IPv6 Address: The IPv6 address of the client the WMR is looking for.
Request Sender WMR IPv6 Address: The IPv6 address of the WMR who sent the request.
Sender WMR CMM Request Sequence Number: The request Client Manager Module sequence number generated by the sender.

2) UCREQ: The Unicast Client REQuest (UCREQ) packet is periodically issued in order to have reachability confirmation, *i.e.*, to confirm that the foreign client is still associated to the same WMR. The packet format is the same as for MCREQ, except for the subtype field which is set to 2, and the fact that it is always unicast.

3) CRREP: The Client Request REPly (CRREP) packet is issued when a WMR receives a MCREQ or a UCREQ for a client that is in its LCTable. The WMR sends back the CRREP as a unicast packet directly to WMR that issued the request. The format is presented in Figure 5, and the content is:

Type: Set to 2, Client Advertisement Mechanism.
Unused: Sent as 0; ignored on reception.
Subtype: Set to 3, Client Request REPly (CRREP).
Requested Client IPv6 Address: The IPv6 address of the client the reply refers to.
Request Sender WMR IPv6 Address: The IPv6 address of the WMR who sent the corresponding client request.
Client Manager WMR IPv6 Address: The IPv6 address of the WMR who sent the reply.
Sender WMR CMM Request Sequence Number: The request Client Manager Module sequence number generated by the sender.

4) CWIT: When there is a client that is not anymore associated with a WMR, the corresponding entry in the LCTable has to be deleted. A unicast Client WIThdraw CWIT packet is sent to each WMR present in the ExternWMRs list, in order to advertise the event. The format is presented in Figure 5, and the content is:

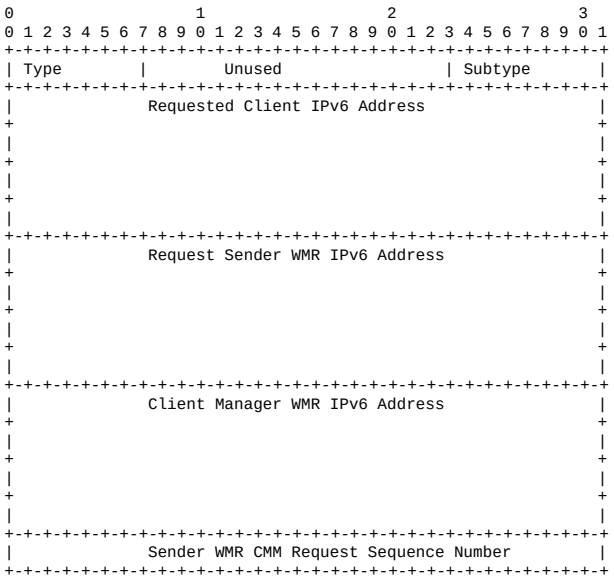


Fig. 5. `CRREP/CWIT/CWREP` Packet format.

Type: Set to 2, Client Advertisement Mechanism.
Unused: Sent as 0; ignored on reception.
Subtype: Set to 4, Client WIThdraw (`CWIT`).
Withdraw Client IPv6 Address: The IPv6 address of the client the withdraw request refers to.
Withdraw Sender WMR IPv6 Address: The IPv6 address of the WMR who sent the withdraw request.
Withdraw Destination WMR IPv6 Address: The IPv6 address of the WMR the request is addressed to.
Sender WMR CMM Sequence Number: The request Client Manager Module sequence number generated by the sender.

5) CWREP: The Client Withdraw REPly (`CWREP`) packet is issued when a WMR receives a `CWIT` for a client that is in its `FCTable`. The WMR updates its `FCTable` and sends back the `CWREP` as a unicast packet directly to the WMR that issued the request. The format is presented in Figure 5, and the content is:

Type: Set to 2, Client Advertisement Mechanism.
Unused: Sent as 0; ignored on reception.
Subtype: Set to 5, Client Withdraw REPly (`CWREP`).
Withdraw Client IPv6 Address: The IPv6 address of the client the withdraw reply refers to.
Withdraw Sender WMR IPv6 Address: The IPv6 address of the WMR who sent the corresponding withdraw request.
Withdraw Destination WMR IPv6 Address: The IPv6 address of the WMR who sent the reply.
Sender WMR CMM Sequence Number: The request Client Manager Module sequence number generated by the sender.

IV. NDP Proxy Module

As mentioned in section II-A, the `MeshDV` demon acts also as NDP-proxy on the *wlanI* interface, by means of a specific module. We detail here this module, before the Client Manager algorithm, in order to give a complete overview of the events that the latter has to handle.

To perform its task, the NDP-proxy module needs to capture ICMPv6 packets, opening an ICMPv6 socket.

Algorithm 1 NDP Proxy Algorithm

```
loop
  repeat
    Wait
  until (New Event Occurs)
  if ( New Event == Neighbor Solicitation Packet Received ) then
    TargetClientIP = IPv6 address of requested client
    Ask the Client Manager Module if TargetClientIP is in LCTable
    if TargetClientIP not in LCTable then
      if (Neighbor Solicitation was sent in multicast) then
        Trigger Client Manager module to send a MCREQ
      else if (Neighbor Solicitation was sent in unicast) then
        Trigger Client Manager module to send a UCREQ
      end if
    end if
  else if ( New Event == Client Manager Module Message Received) then
    CREQResult = Client Manager Module Message State
    if (CREQResult == Client Found) then
      Send a Neighbor Advertisement packet on wlanI
    end if
  end if
end loop
```

It filters out all packets except *Neighbor Solicitation* requests, which need to be exanimed. Details on how to use ICMPv6 sockets and how to filter the messages are in [7], while details about ICMPv6 protocol and messages are in [9] and [8].

The NDP-proxy module waits for an event to occur. An event can be the reception of a *Neighbor Solicitation* packet, through the ICMPv6 socket, or a message from the Client Manager module. If the event is a *Neighbor Solicitation* packet, the IPv6 address of the target client is extracted. If the target client is in the `LCTable`, the client itself will reply directly since it is in the local WLAN; no further action is taken. Otherwise, the NDP module triggers the Client Manager module to send a client request packet. If the *Neighbor Solicitation* packet was a multicast packet, a `MCREQ` request is triggered. Otherwise, if it was a unicast packet, a `UCREQ` request is triggered.

The Client Manager module manages request timeouts and always provides anyway an answer to the NDP module. If the Client Manager module answers that a reply is received, the NDP-proxy sends back a *Neighbor Advertisement* packet. This packet sets the MAC address of the `wlanI` interface of the WMR as the MAC address of the target client. With these settings, future data packets, sent to the target client, will pass through the *wlanI* interface, captured by the Forwarder module, encapsulated, and correctly routed inside the mesh network.

The complete algorithm is described via a pseudo language in Algorithm 1. Note that the NDP-proxy module does not implement an active wait; in the algorithm this action is defined as an active loop only for the sake of clarity.

V. Client Manager Module Algorithm

The Client Manager module has two main tasks: the consistent management of both `LCTable` and

FCTable structures and the client advertisement communication on the mesh subnetwork, through the *wmrI* interface. The module is also responsible for maintaining a local counter named *CMM sequence number*, *i.e.*, the Client Manager Module sequence number. This counter is used whenever MCREQ/UCREQ/CWIT packets are issued. The counter is increased by one each time a request is sent. The CMM sequence number increases robustness, since for all requests, the corresponding reply must carry the same sequence number in order to be accepted. Moreover, it limits broadcast overhead. The MCREQ is a multicast packet, sent to FF02::2 (All link-local IPv6 Routers). All one-hop away WMRs will receive the request and will forward the packet if they are not able to reply to the request. In order to forward an MCREQ request only once, each WMR keeps a per-WMR CMM sequence number, stored in the routing table maintained by the Enhanced DV module. The MCREQ is forwarded only if the requested client is not associated with it and the carried CMM sequence number is higher than the one already stored.

Note that the CMM sequence number is not the same sequence number as described by Perkins *et al.* in [1]. The sequence number proposed in [1] is associated with each destination, in order to avoid loops, and is embedded in each entry advertised by route update messages. This feature remains unchanged. The new CMM sequence number is only used in Client Advertisement packets. Route update messages do not carry the CMM sequence number.

When a WMR receives a request message, it updates the CMM sequence number value and processes the message only if the new CMM is higher than the old value. Otherwise the packet is silently discarded.

The Client Manager module reacts only to four types of events: 1) a request has been received; 2) the NDP-proxy module has made a request; 3) a timeout has occurred; 4) a client has associated/disassociated to the *wlanI* interface.

In the following subsections, we describe how the Client Manager module works. The main loop is not described since it is simply an event handler that calls the right subroutine. Furthermore, since the CWIT sending subroutine consists of just preparing and sending the packet, it is not detailed. For the sake of clarity and simplicity, timeout events are not described; a detailed description can be found in [12].

A. *Packet reception* subroutines

1) MCREQ reception subroutine: When a MCREQ is received, the WMR first checks whether the sequence number is already stored in the routing table. If so, no further action is needed since the received MCREQ is a duplicate. If not, the sequence number is stored and the subroutine checks if the request should be forwarded or replied to. This last decision depends on whether the requested client is associated to its *wlanI* or not. Since any MCREQ retransmission increases the sequence number, if a MCREQ, whose sequence number is already stored, is received, it is considered as a duplicate packet. The subroutine is detailed in Algorithm 2.

Algorithm 2 MCREQ Reception Subroutine

```
if (CMM Request Sequence Number higher than the one stored in RTable) then
  Update the CMM Request Sequence Number in the correct entry of the RTable
  if (Request Client in LCTable) then
    if (Request Sender not in ExternWMRs list) then
      Add Request Sender in the corresponding ExternWMRs list
    end if
    Prepare CRREP Packet
    Send unicast CRREP Packet to Request Sender
  else
    Forward MCREQ Packet on wmrI using FF02::2 address
  end if
end if
return
```

Algorithm 3 UCREQ Reception Subroutine

```
if (CMM Request Sequence Number higher than the one stored in RTable) then
  Update the CMM Request Sequence Number in the correct entry of the RTable
  if (Request Client in LCTable) then
    if (Request Sender not in ExternWMRs list) then
      Add Request Sender in the corresponding ExternWMRs list
    end if
    Prepare CRREP Packet
    Send unicast CRREP Packet to Request Sender
  end if
end if
return
```

2) UCREQ reception subroutine: When a UCREQ is received, the WMR first checks whether the sequence number is already stored. If so, no further action is needed since the received UCREQ is stale or duplicate. If not, the sequence number is stored, and the subroutine checks if the request needs to be replied to. Again, this action depends on whether the requested client is associated to its *wlanI* or not. The subroutine is detailed in Algorithm 3.

3) CRREP reception subroutine: When a reply with a correct sequence number is received, the sender is added in the correct entry of the FCTable list and the NDP-Proxy module is informed. The subroutine is detailed in Algorithm 4.

4) CWIT reception subroutine: When a CWIT request is received, the CMM sequence number is checked to see if the request is a new one. If so, it does not matter if the client has been already withdrawn from the FCTable or not, the WMR replies regardless. Indeed, if a WMR receives a new withdraw request for an already withdrawn client, this means that the previous reply got lost. The subroutine is detailed in Algorithm 5.

5) CWREP reception subroutine: When a withdraw reply with a correct sequence number is received, the sender is deleted from the ExternWMRs list of the

Algorithm 4 CRREP Reception Subroutine

```
if (Request Sequence Number has no match in Pending_Requests_Queue) then
  Discard Packet
else
  Stop corresponding timeout timer
  ClientIP = Target Client IP of the request
  Delete corresponding request packet from Pending_Requests_Queue
  if (Client Manager WMR not in FCTable for ClientIP) then
    Add Client Manager WMR to ManagerWMRIP list for ClientIP
  end if
  Message to NDP-Proxy module ClientIP reply received
end if
return
```

Algorithm 5 CWIT Reception subroutine

```
if (CMM Request Sequence Number higher than the one stored in RTable) then
  Update the CMM Request Sequence Number in the correct entry of the RTable;
  Prepare CWREP Packet;
  if (Withdraw Client is in FCTable) then
    Delete Withdraw Sender from ManagerWMRIP list;
    if (ManagerWMRIP list is empty) then
      Delete Withdraw Client entry from FCTable;
    end if
  end if
  Send CWREP Packet
end if
return
```

Algorithm 6 CWREP Reception Subroutine

```
if (Request Sequence Number has no match in Pending_Requests_Queue) then
  Discard Packet
else
  Stop corresponding timeout timer
  ClientIP = Target Client IP of the request
  Delete corresponding request from Pending_Requests_Queue
  delete Client Manager WMR from ExternWMRs list for ClientIP
end if
return
```

Algorithm 7 MCREQ Send Subroutine

```
if (Requested Client IP is not in FCTable) then
  Create a new entry in FCTable for the requested Client IP
end if
Update CMM_SEQUENCE_NUMBER
Prepare MCREQ Packet
Send MCREQ Packet multicast FF02::2
Append MCREQ to Pending_Requests_Queue
Set timeout counter at CRREP_TIMEOUT
Start timeout Timer
return
```

correct entry of `LCTable`. The subroutine is detailed in Algorithm 6.

B. NDP-proxy requests subroutines

1) MCREQ send subroutine: This subroutine simply prepares the packet and sends it to the multicast address FF02::2, on the wmrI interface. When the packet is created, the CMM sequence number is increased by 1. The subroutine is detailed in Algorithm 7.

2) UCREQ send subroutine: This subroutine is very similar to the `MCREQ` send subroutine, the algorithm is not depicted here. The main difference is that the packet is unicast to the client manager indicated in the entry of the `FCTable`.

C. Client Disassociation subroutines

1) CWIT send subroutine: This subroutine is very similar to the `UCREQ` send subroutines, except that the destination is taken from the `LCTable`. The algorithm is not detailed since it just consists of preparing the packet and sending it on the correct interface.

VI. Forwarder Module

In order to avoid maintaining state information along the WMR-path connecting two clients associated to a different WMR, the `MeshDV` uses a tunneling approach. When a client sends a packet to a remote client, because of the NDP-proxy module, the packet is delivered to the local WMR. The local WMR is now in charge of forwarding the packet. At the IP level, the packet contains the local client address as a Source Address and the remote client address as a Destination address. The kernel is not able to route this packet because it is not aware of remote clients. Nevertheless, the `MeshDV` has all the needed knowledge and can easily create a new IPv6 packet destined to the remote WMR, containing the original one. Since routes between WMRs are setup proactively, WMRs along the path are able to route the new packet without even knowing the destination client address. Only the WMRs where the two clients are associated are aware of the ongoing communication. Indeed, they have the local client and the remote client addresses respectively in the `LCTable` and the `FCTable`.

An alternative solution, in order to route a packet toward the WMR with which a client is associated, is the *Routing Header* option of IPv6. Nevertheless, since WMRs along the path do not know the source client, in case of packet drop, they would try to send an ICMP packet to the source client. But the kernel is unaware where the client is and unable to find it since it has access only to the local *wlanI*. `MeshDV` is able to find the client but it should be aware of all packets dropped by the kernel, turning into a very complex software architecture. Using tunneling makes `MeshDV` simpler. In the following, the operations of encapsulation and decapsulation are described.

Decapsulation. Tunneling needs to be managed by the `MeshDV` demon only when the packet enters the mesh network, *i.e.*, when it comes from the *wlanI* interface and is forwarded on the *wmrI* interface. In the other direction, when a tunneled packet arrives at the destination WMR, the netBSD kernel is able to decapsulate it. Then, since the kernel is aware of the clients associated to the *wlanI* interface, it is able to deliver the inner packet without any additional operation.

Encapsulation. The encapsulation is done inside `MeshDV`, because the kernel is not aware of which WMR the destination client is associated with. All IPv6 packets destined to remote clients that arrive at the *wlanI* interface are directly delivered to the Forwarder module,

in order to be encapsulated. They must not pass through the kernel. For this purpose, on *wlanI*, PF [10] is used to filter them; they are captured at link-level and delivered to `MeshDV` by the Pcap [11] utility. The packet received from the Pcap filter is extracted from the layer 2 frame, a new IPv6 header is pre-pended to it, and finally delivered to the kernel. The kernel is now able to route the packet, through the *wmrI* interface to the correct WMR.

VII. Testbed and Preliminary Observations

The testbed on which the `MeshDV` is currently being implemented is composed of custom WMRs assembled in our laboratory. Each WMR is a Soekris net4521 box, running NetBSD 2.0, and using only the IPv6 protocol stack. The *wlanI* interface is a Proxim 8470-WD b/g card set in HostAP mode. The *wmrI* interface is a NetGate 5354 MP Plus Aries2 4G a/b/g set in ad hoc mode, using 802.11a.

`MeshDV` implementation is not yet completed at the time of this writing, nevertheless, some initial observation can be underlined. The first observation we achieved while implementing `MeshDV` is that global unique addresses are useless on the *wmrI* interface. Indeed, it is possible to reach any WMR by simply using the *wlanI* global address by correctly setting the kernel routing table. Interfaces of WMRs in the mesh subnetwork have a Link-Local unicast address, which is not unique and has a local scope. Thus, taking the topology of Figure 1, the kernel routing table of WMR_2 concerning route to WMR_4 and WMR_1 becomes:[3]

```
Destination                     Gateway                     Flags

default                         FE80::202:6FFF:FE20:F7ED%ATH0  UG
2001:660:3302:2A21::1           FE80::202:6FFF:FE20:F7ED%ATH0  UGH
2001:660:3302:2A21::4           FE80::202:6FFF:FE20:F7ED%ATH0  UGH
FE80::202:6FFF:FE20:F7ED%ATH0   00:02:6F:20:F7:ED              UGL
```

`MeshDV` also performs gateway advertisement. Thus, a default route is always present on each WMR, as shown in the above example. This allows to sligthly reduce overhead, since no special messages need to be issued. Kernel routing table size is always proportional to the number of WMRs. `MeshDV` routing table size is given by the sum of the size of the routing table inside the Enhanced DV module (proportional to the number of WMRs) and the size of the `LCTable` and `FCTable` tables. The sizes of these tables are proportional to the number of clients present in the network and the ongoing communication between clients associated to different WMR. The `FCTable` of the gateway WMR contains the list of all clients of the network. Indeed some basic services, like DNS, are placed outside the mesh, thus, there is always traffic from and to all the clients. In [12] we present in detail the reduction in the routing table size that `MeshDV` can achieve.

Concerning scalability, we expect `MeshDV` to scale like RIPng [6], since the internal routing paradigm is basically the same.

[3] In our testbed the client subnetwork prefix is 2001:660:3302:2A21/64 at which the WMR number is added in order to have the global address of the *wlanI* interface. Furthermore, the MAC address of WMR_4 *wmrI* interface is 00:02:6F:20:F7:ED.

VIII. Conclusion

In this paper, we presented the prototyping research work for the implementation of `MeshDV`, a routing architecture for wireless mesh networks. We described the general functionality of `MeshDV` and detailed its main components. At present, tests are being performed on a testbed deployed at the LIP6 laboratory, aiming to deploy a large WMN covering the campus of the University of Paris VI.

Besides the fact that the protocol is implemented over IPv6, there are other features that make it interesting. `MeshDV` maintains routes between WMRs in a proactive fashion, while searching for routes toward clients in an on-demand fashion. The architectural characteristics of `MeshDV` have been strategically chosen in order to obtain the following advantages: 1) reduced routing table sizes, since there is no need for keeping all routes toward all clients; 2) easy management of clients that change WMR association, since related information is maintained only on edge WMRs.

References

[1] C.E. Perkins, P.R. Bhagwat, "Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers," *in Proceedings of ACM SIGCOMM*, London, sep. 1994.

[2] B. Gupta, "Design, Implementation and Testing of Routing Protocols for Mobile Ad-Hoc Networks", Master Thesis, University of Illinois at Urbana-Champaign, 2002 Online at: *http://decision.csl.uiuc.edu/ wireless/dsdv/*

[3] "LocustWorld Bio Diverse Networking Unleashed!", Online at: *http://www.locustworld.com/*

[4] "MIT Roofnet", Online at: *http://pdos.csail.mit.edu/roofnet/doku.php*

[5] "RFC 2460: Internet Protocol, version 6 (IPv6) Specification", 1998.

[6] "RFC 2080: RIPng for IPv6", 1997.

[7] "RFC 2292: Advanced Socket API for IPv6", 1998.

[8] "RFC 2462: Neighbor Discovery for IP Version 6 (IPv6)", 1998.

[9] "RFC 2463: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", 1998.

[10] "PF: The OpenBSD Packet Filter", Online at: *http://www.openbsd.org/faq/pf/*

[11] "Programming with Pcap", Online at: *http://www.tcpdump.org/pcap.htm/*

[12] L. Iannone, "MeshDV: Implementation Draft", Online at: *http://www-rp.lip6.fr/~iannone/meshDV.pdf*

Posters and Demos

Social networks, novel communication applications and needs in mobile contexts

Claudia Brazzola
University of Applied Sciences of Lugano, Dept. of Social and Administrative Sciences
claudia.brazzola@supsi.ch

Abstract

This paper deals with how some communication technologies and applications intervene in modeling and changing the characteristics of our modern society structure. It starts from the concepts of mobility, of strong and weak ties and concludes by analyzing some examples of novel communication applications with focus on applications for mobile ad hoc networks and by presenting some results from studies about the elderly and new technologies of communication, conducted within the MobileMAN project.

1. Introduction

The aim of this paper is to reflect on communication applications, with particular interest in mobile applications of ad hoc networks and the implications that this has on the structure of our modern society. Some of the content is the result of studies conducted within the MobileMAN project that had the focus on the potential consequences of ad hoc networks on users. The MobileMAN project is a EU project of the Information Society Technologies area that aims to design, implement and validate a new paradigm of ad hoc networks, from the technical, economic and social points of view.

2. Modern Social Networks

One's social network is the complex of relationships with other individuals. Strong ties are those such as kin relations and close personal friends, whereas weak ties are loose acquaintances such as those connections made at a party. Following Granovetter's thought [1] we consider both types of ties as vital for an individual and analyze in what way technology and existing or future applications can help maintain and build both weak and strong ties.

3. Mobility of Individuals – New Needs and New Opportunities

3.1. Internet + Mobility = ?

We start from two central concepts that we are identifiable in our society: the increasing importance of the Internet connectivity and the high mobility of individuals. Internet has been changing the way we live our life. The next element is mobility: numerous workers are commuters and many people are rarely at home. Portability of communication means that individuals are virtually able to communicate and connect to networks anywhere and anytime. They must not wait to be in a particular place [2]. Putting together these two aspects, a whole range of needs result. In one of his articles, Rheingold writes "[...] participants in online communities will remain in continuous contact over multiple platforms on desktops and in mobile devices, and will be used to coordinate group activities in the geographic world, thus blending affinity-based and local-acquaintance-based social communication." [3] He speaks of the combination of the Internet world and the mobile world, with which the user will interact continuously.

3.2. Implications for MobileMAN – New Applications / Scenarios: an Experiment with a Wiki Website

Mobile phones allow people to communicate in many ways such as SMS, MMS, e-mail. However, they do not allow expanding social networks – they only help individuals in managing existing contacts. In fact, "mobile communications are organized around known social networks. People call and message people they already know. Most often, you communicate with people who are already in your address book." [4]. This is also supported by data from a study on mobile phones and ad hoc networks currently ongoing at SUPSI. So far, participants in this qualitative study agree on saying that mobile phones

do not help expanding social networks, but maintaining and developing existing relationships.

We believe that there is an untapped market for applications that allow expanding social networks, more than those applications that enable individuals to manage existing relationships. MobileMAN could exploit this opportunity by developing applications that are oriented to this opportunity. An experiment of participatory design, conducted within the MobileMAN project and involving a number of students of Helsinki University of Technology aimed at developing new applications and scenarios for ad hoc networks through group collaboration either using offline medium (paper) or online medium (wiki website [5]). A wiki is a particular type of site whose content is quickly editable through any browser without the need for the user to have any programming skill. The experiment had also the objective to verify the appropriateness of wiki as a tool for participatory design activities in technological and innovative fields. Students created some scenarios of use for MobileMAN in small groups, and in a second round were asked to complete and comment on any three chosen scenarios among the developed. This exercise was within one course of their curriculum in networking and was not compulsory; who participated received a reward for the course final grade. Participation was not too extensive – 27 students (of a class of 112) worked in the scenario developing activity, and only 20 collaborated until the end of the exercise by also filling in a questionnaire, which covered various topics, from an evaluation of the exercise itself to opinions about the future development of ad hoc and infrastructure-based networks. We interpreted this low participation with the abstractness of the concept of ad hoc. The exercise, in fact, revealed also the difficulty to imagine applications that are exclusive for ad hoc networking devices, most likely because of the abstract nature of the ad hoc concept. Being based on cooperation of the users, it relies on a completely new model of low-level architecture. Applications that run on top can be the same as those that run on mobile phones or others. The hypothesis is that innovative applications that empower the user to expand their social network will be more interesting and appealing. Ad hoc networks are infrastructure-less and consequently (ideally) with no use cost for the user. This, however, cannot be the only driving motive that pushes for adoption – especially if the quality of service is not as high as is now the quality of mobile phones applications (voice clarity, network availability). In this vision, applications can play a fundamental role in the adoption process of the technology. We present the summarized content of two scenarios that are particularly valuable, developed by two groups of students.

- **Public festival scenario** – a public festival is an occasion where there are many people gathered in a limited area and it can happen that base stations are stuck because of overload. In such situation ad hoc would be helpful. In the developed scenario, three boys use their MobileMAN device to micro coordinate themselves and to receive real-time information about facilities. Another application in this scenario is a profile manager that alerts the user if there is in close vicinity some user that matches their profile. This is an application designed for extending social networks. Although some aspects of this scenario need further analysis of realistic realization it provides interesting ideas to work on.
- **Where to party scenario** – big cities often offer far too many options to spend free time and it may be difficult to make a choice. This application would combine location information, by informing of opportunities in the user nearby, social networking expansion, by viewing the structure of the network, with for example, friends of their friends, and the digital equivalent of "word of mouth", that is, MobileMAN users post information and broadcast it to the network of persons they are connected to.

SMS has been the killer application for mobile phones with millions of messages sent every day. However, chat-SMS[1] have not so far had great success because "SMS connects people very inefficiently. Those who design future services would do well to search for more efficient ways of connecting people." [6] The last example of scenario could be a response to this need for more efficient applications of sending messages.

3.3. MobileMAN Whiteboard

Within the MobileMAN project, a peer-to-peer multicast application called whiteboard is being developed. It can be considered as a first version of a distributed chat platform for text writing or graphic drawing. Comparing this application with traditional SMS written and sent by mobile phone, we can say that it is certainly interesting, since it places in between SMS and chat. Being a multicast system, it would

[1] Chat-SMS is a system where users subscribe to a chat service (channel) and receive each message sent by all other participants to the channel. The dubious issue in this service is that "it remains to be seen how willing the participants in the chat groups are to pay for EVERY message sent to the chat channel". [7]

increase the limited efficiency of SMS – something that is in line with Saarikoski's thought [6].

3.4. The Elderly – the "Grey Digital Divide"

The elderly constitute a particular target when defining applications and services for specific groups: they have a range of needs that at the moment are unaddressed. MobileMAN expressed the intention to privilege this group of individuals and potential end users of MobileMAN, based on the belief that technology can help elderly people improving their life. Therefore, a study of their relationship with new ICTs is currently being carried out at SUPSI within the project. The study focuses on technologies like TV, VCR, DVD, computer and internet, mobile phone and telecare wristalarm[2]. All participants had a telecare wristalarm installed and were interviewed about their personal and social situation, existing relationships and activities and technology they use in their everyday life. 10 out of 11 interviewed lived alone (which was the reason why they adopted the telecare wristalarm). The initial idea was that for elderly people living alone, new ICTs would have been particularly valuable as they would have allowed a higher communication and interaction with society, even if physically impaired (difficulty to go out and walk e.g.). However, findings demonstrated that the elderly have a very low interest in new ICTs, even if these can help them overcoming their isolation, which is – as said – very often a characteristic of their social and personal situation. As a consequence, it is very difficult to involve this category of people in participatory design activities within MobileMAN. We explained this disinterest towards innovative communication technologies (mobile phone, internet, chat) as related to their particular life path: they lived through war time, and lived generally without all the communication means we dispose of nowadays. This belonging to a different world is very clear in the fact that almost all the interviewed elderly expressed the inability to understand the need for communication and therefore considered as useless communication empowering technologies (chat, email, internet in general, mobile phones). This result confirms findings of Millward [8], who researched the issue of "grey digital divide". This way of thinking about new ICTs, however, is probably only typical of this generation of elderly: in 15-20 years the old people will have a different relationship and opinion about them. The "grey digital divide" might diminish naturally with time.

[2] This is a system created for elderly who live alone and are at risk of falling or have health problems. It is wrist button that can be pushed in case of necessity and that contacts directly the emergency station, which deals with the situation. This system has proven very useful since there is no time loss due to the inability of the person to reach the phone in case of emergency.

4. Conclusions

In this paper we have started from the networked structure of society. We presented two scenarios developed by two groups of students within an activity of the MobileMAN project, one application (whiteboard) currently being implemented by MobileMAN partners that can provide some interesting ideas for novel applications to be developed for ad hoc network devices. Moreover, we presented some results of a study conducted with a number of elderly people on their relationship with new communication technologies. The main results confirm the existence of a "grey digital divide" that might however diminish with time.

5. References

[1] M. Granovetter "The Strength of Weak Ties: A Network Theory Revisited", *Sociological Theory Vol.1*, New York, 1983, pp. 201-233

[2] B. Wellmann, "Physical Place and Cyberplace: The Rise of Personalized Networking", *International Journal of Urban and Regional Research 25*, Special issue on "Networks, Class and Place" ed. Talja Blokland and Mike Savage, Toronto, 2001

[3] H. Rheingold, "Mobile Virtual Communities", 2003, published on www.thefeature.com

[4] H. Rheingold, op.cit.

[5] http://mobileman.projects.supsi.ch/wiki/index.php

[6] Saarikoski quoted in H. Rheingold, "Email, Scale-Free Networks, and the Mobile Interne", 2005

[7] www.m-indya.com/mwap/sms/user_appl.htm

[8] P. Millward, "The 'grey digital divide': Perception, exclusion and barriers of access to the Internet for older people", 2003, published on www.firstmonday.dk

6. Acknowledgments

Many thanks to Prof. Dr. Christian Marazzi and Dr. Jennifer Duyne for reading this article and for their precious comments and suggestions.

On the Dimensionality of Wireless Connectivity Traces

George Roussos
Birkbeck College
University of London
g.roussos@birkbeck.ac.uk

Abstract

Routing efficiency in wireless networks can be greatly improved by matching mobile host connectivity patterns. To this end, over the past few years considerable effort has been invested in developing predictors of mobility patterns that is, models of mobile host movement so that for a specific sequence of recorded locations to predict the most likely subsequent location. In this paper, I initiate a study of the structure of the connectivity space itself. I analyze samples from the Dartmouth data set[1] and conduct an exploratory study of the structure of the underlying space. In particular, I compute the singular values of the time-weighted connectivity matrix, and relate this result to principal component analysis. Initial findings indicate that the degrees of freedom of the space induced by the connectivity matrix is very low with respect to the number of access points and mobile hosts involved; that the dimensionality of the underlying space grows slowly with the size of the sample; and, that the distribution of its eigenfrequencies follows a power law.

1. Introduction

Wireless networks can better serve mobile hosts by employing client location information to better anticipate connectivity patterns. Predicting accurately the location of hosts can potentially improve the performance of wireless routing and the robustness of the network infrastructure itself, thus improving the user experience for a variety of applications. These improvements lead to a better user experience, to a more cost-effective infrastructure, or both. As a result, during the past few years a number of location predictors have been proposed in the literature based on a variety of complementary techniques including Markov-based, compression-based, PPM, and SPM mechanisms. Such approaches infer models of mobile host movement patterns so that for a specific recorded sequence of recorded locations to predict the most likely subsequent location. A comprehensive comparative evaluation study of the relative performance of several of these algorithms on the Dartmouth mobile data trace [2] including a detailed description of their structure and performance can be found in [4].

In this poster, rather than focusing on predictors I initiate the study of the structure of the connectivity space itself so as to understand its core characteristics. To do this, I also employ the Dartmouth data trace to reconstruct the time-weighted connectivity matrix between access points and mobile clients which we use as the basis of this investigation. Several techniques are employed to explore the structure of this space, with a view in all cases to identify the existence of a small subset of components or eigenfrequencies that characterize accurately the client connectivity behavior. Effectively, I aim to determine a basis for a low-dimensional projection of the connectivity matrix that provides an appropriately accurate approximation to the overall connectivity patterns.

2. SVD and Principal Component Analysis

Let $L = \{l_{ij}\}$ be the $m \times n$ connectivity matrix of a data trace defined by setting the element l_{ij} to be

[1]Many thanks too David Kotz and other members of the Dartmouth Centre for Mobile Computing for providing access to this data.

proportional to the time mobile host j is connected to access point i, for a data sample that includes traces of n hosts and m base stations. Note that row i of L describes the time each sighted client has spend connected to base station i, and column j of L describes the time spend by host j connected to each base station. Hence, I refer to the row vectors of L as the connectivity profile for access point i and to the column vectors as the connectivity pattern of client j. Due to the rather limited mobility of hosts in the Dartmouth traces, the matrix L is sparse.

The singular value decomposition (SVD) of L is

$$L = USV^T, \tag{1}$$

where U is an $m \times n$ matrix, S is an $n \times n$ diagonal matrix, and V also an $n \times n$ matrix. The columns of U are called the left singular vectors and form an orthonormal basis for the connectivity patterns of clients, that is $u_i u_j$ is zero except where $i = j$ when it is one. The rows of V^T contain the elements of the right singular vectors and form an orthonormal basis for the connectivity profiles for the access points.

The matrix S is zero everywhere except at the diagonal, that is $S = \mathrm{diag}(s_1, \ldots, s_n)$, where it contains the so-called singular values s_k of L. Singular values are ordered so that the highest singular value is in the upper left index of the matrix S. Note that if L is square, that is $m = n$, then the SVD is equivalent to the solution of the eigenvalue problem.

With the SVD at hand, we can compute the closest r-rank matrix to L as follows

$$L^{(r)} = \sum_{k=1}^{r} u_k s_k v_k^T \tag{2}$$

so that $L^{(r)}$ minimizes the sum of the squares of the difference of the elements of L and $L^{(r)}$. Standard approaches to compute the SVD can be found in [3].

2.1 Relation to principal component analysis

Principal component analysis (PCA) captures the variance in a dataset in terms of its so-called principle components. The SVD is intimately related to PCA when principal components are calculated using the covariance matrix. If each column of L is centered then $L^T L$ is proportional to the covariance matrix of the connectivity profile for the access points. Moreover, diagonalisation of $L^T L$ yields V^T and thus the principal components of the connectivity profile. in other words, the right singular vectors of L are the same as the required principal components. Further, the eigenvalues of $L^T L$ are the singular values of L, which are proportional to the variances of the principal components. Overall, the matrix US contains the principal component scores, which are the coordinates of the connectivity profile in the space of principal components. If instead of centering the columns of L we center its rows then $L^T L$ is proportional to the covariance matrix of the connectivity patterns of the mobile clients. Similar to above, the left singular vectors are also the principal components of the connectivity pattern space; the singular values are proportional to the variances of the principal components; and the matrix SV^T contains the principal component scores, that is the coordinates of the connectivity patterns in the space of principal components.

2.2 Results

In this section I report on preliminary results of the analysis of connectivity patterns using the Dartmouth mobile data trace. Logfiles provided by the CMC were post-processed to extract the connection/disconnection patterns of particular hosts to particular base stations. Several data sets where developed consisting of up to $90,000$ samples, which were subsequently analyzed following the discussion in previous sections.

As noted earlier the main focus of the analysis is to better understand the underlying structure of this space which in this case is characterized by the range of the singular values. For the largest sample we find that the majority of the singular values are relatively small with respect to the largest components, in fact only 4 of those are within 10% of the magnitude of the largest one. Figure 1 provides some more information regarding the actual distribution of the spectrum: the magnitude of the computed singular values appears to follow a power law distribution.

Both observations point towards the fact that it should be possible to reconstruct the full connectivity matrix using only a small number of principal components within a high accuracy. Indeed, using only the 29 principal components it is possible to achieve accu-

racy of the order of 0.1% for this sample. Thus, the variability of the connectivity patterns in the data set is very low and can be predicted very well using a subspace of very low dimensionality.

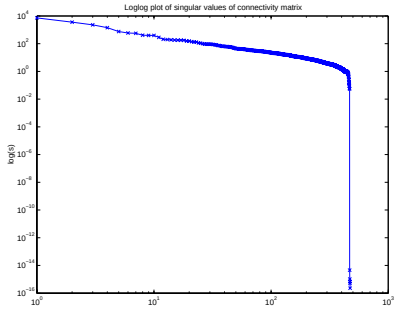


Figure 1. Logarithmic plot of the singular values of the connectivity matrix by size ($90,000$ **samples).**

3. Discussion and Conclusions

Several papers on wireless networking measurement observe that mobility patterns of individual hosts are likely to contain a considerable amount of periodicity. The underlying reasons for this are mainly economic and demographic, and dictate that clients move within a small range of different velocities, and travel along similar routes with most journeys starting and ending at similar places. As a consequence, there is only a small number of approximately discrete frequencies characterizing the behavior and the connectivity profile of the mobile station (and its user). In this paper I provide some preliminary results regarding the dimensionality of the space induced by the mobility patterns observed within a particular experimental setting. Moreover, some early results are highlighted regarding the number of degrees of freedom that exist in the data and have estimated the order of accuracy of reconstruction using a low-dimensional approximation.

This evidence can be used to potentially improve the performance of wireless routing and the robustness of the wireless network infrastructures. For example, using the computed principal components it is possible to pinpoint bottlenecks in wireless networks as well as loss points for example due to interference. Knowledge of such locations allows to locate additional resources where they are needed to improve reliability for example hop-by-hop rather than end-to-end packet loss recovery.

More importantly, these findings provide evidence in support of a recent conjecture by Jon Crowcroft [1] regarding the feasibility of a common network architecture that overarches both mobile wireless mesh and fixed networks. He observes the following correspondence between wireless mesh and wireline fixed networks:

Mesh network		Wireline network
Mobility	$\Longleftrightarrow$	Buffering
Freq. distribution		Router out degree

A critical element for the proposed unified reference model is that the distribution of journey frequencies follows a similar pattern to the distribution of communication popularity, as recorded in the out degree of the connectivity graph of Internet routers for example. This would imply that the routing system for both types of networks would have similar properties.

In this paper I provide a strong indication that this is indeed the case, since the computed singular values of the connectivity matrix or else the journey frequencies clearly appear to follow a power law distribution (cf. Figure 1) and is thus qualitatively similar to the router out degree on the Internet.

References

[1] J. Crowcroft. Communication to the UCL Mobile Systems Group. 10 September 2004.

[2] D. Kotz and K. Essien. “Analysis of a Campus-wide Wireless Network”. *Wireless Networks*, 11:115-133, 2005.

[3] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[4] L. Song, D. Kotz, R. Jain, X. He. “Evaluating next-cell predictors with extensive Wi-Fi mobility data”. *Proc. 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*:1414-1424, 2004.

Cross-Layer Support for Group-Communication Applications in MANETs*

Marco Conti, Franca Delmastro
CNR, IIT Institute
Via G. Moruzzi, 1 – 56124 Pisa, Italy
{firstname.lastname}@iit.cnr.it

Jon Crowcroft, Andrea Passarella
University of Cambridge, The Computer Laboratory
15 JJ Thomson Avenue – Cambridge CB3 0FD, UK
{firstname.lastname}@cl.cam.ac.uk

Abstract

P2P systems are a natural way of supporting group-communication applications in MANETs. In this paper we discuss our experiences in developing such an application in the real world. We highlight limitations of legacy P2P systems, and show that solutions based on cross-layer optimisations are very promising.

1 Introduction

One of the most interesting class of applications that can be envisaged for MANETs is represented by group-communication applications. In the framework of the MobileMAN Project [6], we are investigating the viability of developing such kind of applications on real ad hoc networks. To this end, we developed the Whiteboard application (WB), which implements a distributed whiteboard among MANET users. WB usage is very intuitive (see Figure 1). Each MANET user runs a WB instance on her device, selects a topic she wants to join, and starts drawing on the canvas. Drawings are distributed to all nodes subscribed to that topic, and rendered on each canvas. We believe that these simple, "Plug&Play" applications will be of great value for MANET users.

Developing this kind of applications in MANETs is a challenging task. In this paper we present the networking solutions we have studied and tested to this end. We present alternative networking frameworks for supporting WB-like applications (Section 2). Then, we compare a standard P2P system (Pastry [8]) with CrossROAD [5], the P2P system optimised for MANETs that we have designed within these frameworks (Section 3). Advantages of the CrossROAD approach are presented by means of experimental results in Section 4. Finally, Section 5 concludes the paper.

2 WB integration in MANETs

Group-communication applications such as WB are distributed, self-organising, decentralised in nature. Designing them on top of P2P systems guarantees a great flexibility and optimised performances exploiting P2P policies to distribute and recover information. Figure 2 depicts the abstractions we have used to support WB. The network level provides basic connectivity among nodes through IP-like routing and transport protocols. On top of them, a structured overlay network, comprising nodes that participate in the WB application, is built. The overlay abstraction is the fundamental substrate for any P2P application, providing functionalities such as logical node addressing (instead of topological, IP-like addressing) and subject-based routing. Finally, an additional multicast level is used to efficiently distribute contents generated by application users to all nodes in the overlay. These abstractions make quite straightforward develop group communication applications. They hide the complexity of low-level communications, group management, and data distribution, and provide a robust, flexible, self-organising networking environment.

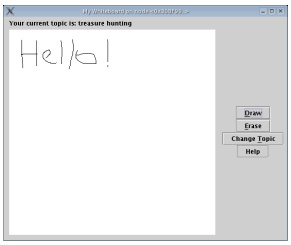


Figure 1. The WB application interface

Figure 3 shows the complete networking solutions we have used to support WB in real-world MANETs. We have defined a first architecture (referred to as *legacy*), that uses state-of-the-art components to implement the abstractions in Figure 2. Specifically, it uses either AODV [1] or OLSR [7] at the network level, Pastry [8] at the middleware level, and Scribe [2] at the multicast level. While AODV and OLSR represent standard models for ad hoc reactive and proactive routing protocols, Pastry and Scribe have been designed for wired networks. The evaluation of the "legacy solution" indicates weaknesses of these components, and ways to improve them. In order to optimize the entire system performances, a *cross-layer* architecture, as depicted on the right-hand side of Figure 3, has been proposed in [4]. Specifically, the NeSt module allows cross-layer interactions between protocols at different layers. To this aim, NeSt provides well-defined interfaces and data abstractions to protocols [4], joining the advantages of cross-layering

*This work was partially funded by the FET-IST Programme of the European Commission, IST-2001-38113 MOBILE-MAN project.

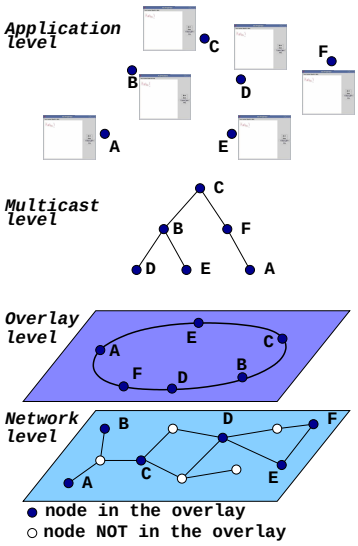


Figure 2. Abstractions supporting WB

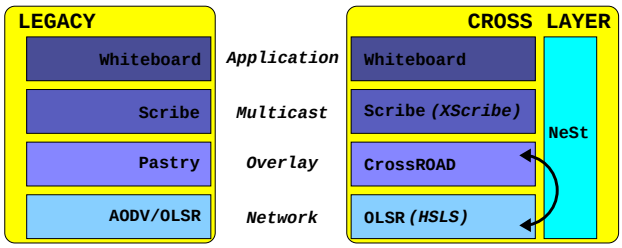


Figure 3. Network solutions: legacy (left) and cross layer (right)

and the scalability of traditional layered approach. CrossROAD represents an optimised solution at the middleware layer that exploits cross-layer interactions with a proactive routing protocol (OLSR in this case) in order to optimize the creation and management of the overlay network. In this paper we do not discuss any other MANET-optimised solutions that could be integrated into the cross-layer architecture. However, in the framework of the MobileMAN project, other such components both at the routing level (Hazy Sighted Link State [9]), and at the multicast level (X-layer Scribe) are being studied and currently under development.

3 Pastry vs. CrossROAD

Pastry represents the P2P computing model on which CrossROAD has been designed to obtain great optimisations on ad hoc networks. It generates an overlay network by organising nodes in a circular logical address space (ring). Specifically, it assigns to each node a *logical* identifier by hashing, for example, the node IP address. Logical identifiers determine the node position in the ring. In addition, messages are routed over the ring by following a subject-based policy, rather than a topology-based one. Specifically, an application wishing to send a message m has to provide a key k linked to m. The k value is hashed

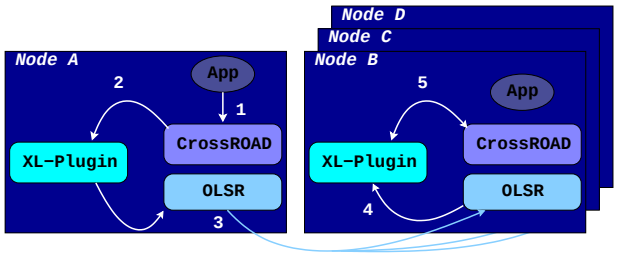


Figure 4. Cross-layer interactions between CrossROAD and OLSR

to obtain an identifier in the same space of nodes' logical ids that is used to select the best destination for that message. The subject-based routing of Pastry sends the message to the node in the ring whose id is numerically closest to the key hashed value. This policy represents the basis for several distributed services; for example, Scribe exploits subject-based routing to build and maintain multicast distribution trees.

To implement subject-based routing, Pastry builds at each node a middleware routing table storing a subset of other nodes' ids. This table is initialised (during a bootstrap phase) and updated (periodically) by exchanging information with the other nodes. When adopted in MANETs, this approach generates quite a lot of network overhead. CrossROAD [5] provides the same Pastry functionalities through the P2P commonAPI [3], but it drastically reduces the overlay management traffic by exploiting cross-layer interactions with a proactive routing protocol. Specifically, CrossROAD implements a Service Discovery protocol, that exploits the broadcast flooding of routing packets to distribute services information. An example of cross-layer interaction between CrossROAD and OLSR is shown in Figure 4. Each application running on CrossROAD has to register itself by specifying a *service id* (step 1). The list of service ids registered at the local node (Node A in the figure) is maintained by the Cross-Layer Plugin (XL-Plugin), which can be seen as a portion of the NeSt module (step 2). The XL-Plugin embeds the list of local service ids into periodic Link-State Update packets generated by OLSR (step 3). On the other nodes of the network (nodes B, C, D in the figure), upon receiving LSU packets containing such list, the routing level notifies XL-Plugin to store the list in its internal data structures. This way, each CrossROAD node has a complete knowledge of all the other nodes providing the same service in the MANET, and it is able to autonomously build the overlay network without generating any further management traffic (step 5). Furthermore, in case of topology changes, the status of the overlay network converged as quickly as the routing protocol does.

4 Experimental Results

The networking solutions described in Figure 3 have been implemented and tested in a read-world multi-hop ad

hoc network. Specifically, the testbed consisted of 8 homogeneous laptops, out of which 6 run the WB application, and the remaining 2 were used just as routers. Experiments that have been run, which mimic the behavior of WB users concurrently drawing strokes on their canvas. Users are represented by software agents that continuously interleave active phases (during which they draw a burst of strokes), and idle phases (during which they just receive others' bursts). Idle phase durations and burst sizes are exponentially distributed. A traffic load of 100% is defined as the load generated by a user drawing – on average – 1 stroke per second.

Due to space constraints, we cannot provide here detailed measurements. Therefore, we discuss the outcomes of some selected experiments, that allow us to highlight several benefits introduced by CrossROAD[1]. Table 1 shows the aggregate throughput (in the sending and receiving directions) of each node during the Pastry 80% and CrossROAD 100% experiments, respectively[2]. These results account for the traffic generated from the routing up to the application level. We mark node C as "C(R)" since it was the root of the Scribe tree. Finally, the last two rows show the average throughput computed over the nodes running WB including and excluding C, respectively. Overall, when CrossROAD is used instead of Pastry, the throughput is drastically reduced. The average value over all nodes in the CrossROAD setup is about one third of the average value in the Pastry setup. It should be noted that, due to Scribe mechanisms, the root node has to handle a far greater amount of application-level traffic than other nodes. Therefore, the throughput reduction due to CrossROAD can be better emphasised by focusing on the last row of the table. If we exclude node C, the average throughput in the CrossROAD setup is about *one fourth* of the average throughput in the Pastry setup. Finally, we found that CrossROAD also improves the stability of the Scribe tree. Table 2 shows the number of sub-trees that are generated in Pastry and CrossROAD setup, respectively. When Pastry is used, the Scribe tree is often partitioned in several isolated sub-trees, resulting in nodes to be isolated from the rest of the network. Instead, this misbehavior is always avoided when CrossROAD is used. It can be shown that it is a byproduct of the Pastry network overhead and bootstrap procedure.

Node	Pastry	CrossROAD
A	16529	2966
B	21278	5069
C(R)	48542	21146
D	29066	7819
E	18047	5993
F	14964	4313
avg	24738	7884
avg (no C)	19977	5232

Table 1. Throughput (Bps) in the Pastry 80% and CrossROAD 100% setup.

Load	Pastry	CrossROAD
20%	1	1
50%	2	1
80% (100%)	3	1

Table 2. Number of sub-trees at the Scribe level.

5 Conclusions

In order to support P2P group-communication applications in MANETs, legacy network architectures designed for *wired* networks are not the real solution. Specifically, such solutions require too much management traffic, and tend to saturate the scarce MANET resources. Optimising the network stack components through cross-layering is a very promising way. In this paper, we have highlighted drastic performance improvements by replacing Pastry with CrossROAD, a cross-layer optimised P2P substrate. Further improvements might be envisaged if also the other P2P components (e.g., Scribe) are optimised according to the cross-layer paradigm.

[1] In the Pastry case, we herafter show only results from OLSR experiments, since OLSR generally allowed to achieve better performances than AODV.

[2] We were not able to run Pastry experiments at 100% traffic load, because the testbed crashed due to excessive network load.

References

[1] AODV, Dept. of Information technology at Uppsala University (Sweden), http://user.it.uu.se/ henrikl/aodv/.

[2] M. Castro, P. Druschel, A-M. Kermarrec and A. Rowstron, "SCRIBE: A large-scale and decentralised application-level multicast infrastructure", IEEE Journal on Selected Areas in Communication (JSAC), Vol. 20, No, 8, October 2002.

[3] F. Dabek and B. Zhao and P. Druschel and J. Kubiatowicz and I. Stoica, "Towards a common API for Structured Peer-to-Peer Overlays", Proc. of the the 2nd International Workshop on Peer-to-peer Systems (IPTPS'03), Berkeley, CA, Feb. 2003.

[4] M. Conti, G. Maselli, G. Turi, "Design and evaluation of a flexible cross-layer interface for ad hoc networks", Proc. of the Fourth Annual Mediterranean Ad Hoc Networking Workshop (MedHocNet 2005), June 2005.

[5] F. Delmastro, "From Pastry to CrossROAD: Cross-layer Ring Overlay for Ad hoc networks", in Proc. of Workshop of Mobile Peer-to-Peer 2005, in conjuction with the PerCom 2005 conference, Kauai Island, Hawaii, Mar. 2005.

[6] "Mobile Metropolitan Ad hoc Network (MobileMAN)", IST-2001-18113 Project, funded by the EC FET-IST Programme, `http://cnd.iit.cnr.it/mobileMAN/`

[7] OLSR, Andreas Tonnesen, Institute for informatics at the University of Oslo (Norway), http://www.olsr.org.

[8] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems", Middleware 2001, Germany, November 2001.

[9] C.A. Santivanez, I. Stavrakakis, and 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.

Real Life Experience of Cooperation Enforcement Based on Reputation (CORE) for MANETs

Claudio Lavecchia, Pietro Michiardi, Refik Molva[1]
Institut Eurecom, 2229, Route des Crêtes
06560 Valbonne Sophia Antipolis
{first name.last name}@eurecom.fr

Abstract

Cooperation enforcement in mobile ad-hoc networks has become a hot topic within the scientific community. Entities belonging to a mobile ad-hoc network are prone to selfishness because being cooperative and participating to basic network functions such as routing and packet forwarding involves resource consumption for the benefit of others. Different approaches have been proposed to promote cooperation in such environments. An implementation of the cooperation enforcement mechanism named CORE [5] is presented in the following as well as a demonstration of its usage on a MANET testbed.

1. Introduction

The hype of trust establishment schemes that the research community is witnessing in recent years results in a proliferation of such mechanisms that target various issues rising at different layers of a communication system.

A particular instance of trust establishment schemes is represented by reputation mechanisms, in which the trust metric takes the form of a reputation measure associated to each entity taking part in a digital transaction. Reputation can be defined as the level of trust inspired by entities based on observations made on entities' past behavior. Intuitively, reputation can be thought of as a metric that drives and regulates the formation of dynamic communities that shares interests and have common goals.

A typical setting in which reputation schemes are used to regulate the formation and the survivability of digital communities is represented by peer-to-peer (P2P) file sharing systems. In P2P communities, reputation can be used to baffle greediness and selfishness of peers that make and use the system while at the same time suffer from the dilemma of constrained resources. Indeed, is there a reason to assume the volunteer participation to the community welfare if no countermeasures are in place to stimulate a fair distribution of the costs incurred by each individual to the community operation? In general the answer is negative, as it has been demonstrated by recent studies [2, 3]. Another interesting domain of application of this type of trust establishment schemes is offered by the mobile ad hoc networking paradigm. In mobile ad hoc networks (MANET), node participation to basic networking functions such as routing and packet forwarding is of fundamental importance. Recent studies [4] show that network performance can be severely degraded even when only a small fraction of the nodes that are part of an ad hoc network deny participation to the network operation. Again, scarce resources are at the origin of a selfish node behavior whereby nodes (and end users operating those nodes) do not want to share the (energetic) costs incurred by the network operation for the benefit of others.

In this paper we focus on a reputation system used to stimulate node participation to the execution of the packet forwarding function in MANETs. We present an overview of the CORE [5] reputation system architecture from an implementation point of view and detail the demonstrative setting in which we carried out the proof-of-concept validation of CORE. The reader should refer to [6] for a detailed description and analysis of CORE.

2. CORE System Architecture

The CORE reputation system uses the watchdog mechanism [7]. A watchdog can be defined as a software component installed on the nodes of a network with the aim of observing neighboring nodes behavior with respect to participation to basic network functions. The solution proposed hereafter addresses only the packet forwarding function and has been implemented and tested on a MANET testbed made of nodes relying on off-the-shelf 802.11b hardware. A MANET node implementing the watchdog mechanism must be able to overhear all the packets that are sent within its wireless channel. To do so, the 802.11b WLAN adapter needs to be operated in the so-called promiscuous mode. This functionality is implemented at WLAN adapter firmware and driver level and enables the WLAN adapter to pass all the packets received to upper layers for further processing. In our MANET testbed we use Dell TrueMobile 1150 WLAN adapters that are operated by the Orinoco driver which works in promiscuous mode out of the box. Once the WLAN adapter is set in promiscuous mode, the packets are captured using the pcap [8] C libraries. Those libraries are available for Windows OS as well, making the porting effort of the CORE mechanism to Windows OS acceptable. The CORE reputation system has been implemented as a Linux daemon, the implementation architecture is illustrated

[1] This research was partially supported by the Information Society Technologies program of the European Commission, Future and Emerging Technologies under the IST-2001-38113 MOBILEMAN project and by the Institut Eurecom.

in Figure 1. Here follows the detailed description of the modules that compose the system:

Sniffer Module: monitors the packets that pass across layer 2 of the TCP/IP stack. This module passes the relevant fields of packet headers to the analyzer module for further analysis in the form of packet descriptors.

Analyzer Module: Receives packet descriptors from the sniffer module and analyzes those descriptors to deduce whether the neighbors are being cooperative or not.

The analyzer module includes an expectation table. Packet descriptors that correspond to packets for which forwarding is expected by a neighbor are stored in this table. The scheduler included in the analyzer module triggers a timeout each time that a packet descriptor is written in the expectation table. Upon timeout expiration on a packet descriptor, the analyzer verifies if the corresponding packet has been forwarded from the neighbor to the next hop. In such case it deduces that the neighbor that forwarded the packet has been cooperative and a positive observation is passed to the reputation module. If the packet has not been forwarded before the timeout expiration, the node that was expected to forward the packet is suspected to be selfish and a negative observation is passed to the reputation module. The analyzer module features an ARP interface that is needed to perform some basic neighbor discovery functions.

Reputation Module: In the original version of CORE [5], the reputation value associated to a node is evaluated in a sophisticated way. The interested reader should refer to [6] for a detailed description and analysis of advanced reputation evaluation functions. For sake of simplicity and in order to provide a proof of concept evaluation of CORE, the real life implementation of our cooperation enforcement mechanism is based on a simpler reputation function described hereafter. The reputation module uses a weighted average function to calculate reputation values for neighbors according to observations provided by 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.

In the current implementation the reputation function is given by:

$$R_K(a) = \sum_{k=0}^{k=B-1} W_k \frac{Obs_a(K-k)}{B} \quad (1)$$

where:

a is the node that is being observed by the watchdog.

B is the number of observation that the node that executes the watchdog keeps in its local observation buffer.

W_k is the weight given to the k-th observation.

K is the actual absolute discrete time.

$Obs_a(K-k)$ is the value of the observation at time $K-k$

Possible observation values are: (+1) if the watchdog detects a cooperative behavior, (-1) if the watchdog detects a selfish behavior.

Punishment Module: Punishes selfish neighbors by denying packet forwarding through the "iptables" Linux framework [9]. The proposed architecture has the advantage of identifying clear interfaces among the system modules, thus allowing the interoperability of the watchdog module with other systems that calculate and exploit the reputation of other nodes, such as for example the one proposed in [7]. Another advantage of this solution resides in the very limited network overhead introduced by its operation: the only additional traffic is a pair of ARP request/reply generated each time that a node running the watchdog analyzes packets that involve a previously unknown neighbor node.

3. CORE Demonstration

The implementation of CORE has been integrated on our MANET testbed. A demonstration has been developed to show the system behavior.

The MANET testbed is composed of 4 nodes, equipped with a WLAN adapter. Two of the nodes are laptops running Windows OS, the third one, where the watchdog software is executed is a laptop running Linux OS, the fourth node can be either a laptop or a Compaq iPaq PDA, in both cases it runs Linux OS.

The demonstration objectives are:

- To show how the CORE system maintains a reputation state for all the neighbors of a MANET node. Two states are possible for a neighbor: selfish and cooperative.
- To show how the economical approach that drove the CORE mechanism design effectively motivates a user that is leaning toward selfishness to reconsider his objectives and be cooperative.
- To show the effectiveness of inherent reintegration mechanism proposed by CORE.

The MANET testbed nodes are logically disposed in a row. We assume the existence of bidirectional wireless links between neighbor nodes. Nodes are disposed as follows:

A↔W↔S↔B

Ideally the physical distances between two non-neighboring nodes of the network are larger that the WLAN card transmission range. Two non-neighboring nodes of the network that wish to communicate will pass through an intermediate node that acts as a router. In reality, the WLAN cards transmission ranges are so that the physical separation in an indoor environment is achieved only when two non-neighboring nodes are more than some tens of meters far from each other and this is quite hard to handle in a demo environment. For this reason we use some "tricks" to logically separate two non-neighboring nodes. Those tricks involve the usage of firewalls or Linux "iptables" framework. Node W is the node that runs the watchdog. This node observes the behavior of its neighbors. According to the observations, node W maintains for each neighbor a reputation state. State for a neighbor can be either cooperative or selfish. With respect to the equation (1), the

watchdog parameters have been set as follows: $B=4$, $W_i=0.25$ (all the weights are equal).

Node S is the node that is operated by the selfish user. As a selfishness model, we assume that the administrator of this node stops forwarding traffic originated by nodes A or W and directed to node B and vice versa when the battery level of the its device falls below a given threshold.

We assume a non-selective selfishness model that is implemented by blocking the forwarding of other nodes traffic through the usage of Linux "iptables" framework.

Node B serves the HTTP connection requests coming from node W while node A acts as an FTP server for connection requests coming from node S.

During the demonstration, node S changes its state from cooperative to selfish. As a result of this change of attitude, node W HTTP connection requests to node B will fail.

CORE console running on node W shows the observations performed by the watchdog on the neighbors' behavior. When node S becomes selfish, a line appears on the CORE console to show the neighbor change of attitude. As soon as this transition is detected, the watchdog issues a punishment request to the CORE punishment module. Node S is immediately punished by node W, which stops forwarding node S packets. From this moment on, the FTP connection requests from node S to node A will stall.

When node S realizes that its FTP connection requests are blocked by node W, it decides to become cooperative again. This behavior transition is again captured by the watchdog on node W and shown on the CORE console. Node W immediately restarts forwarding node S packets. From this moment node S is reintegrated in the network and its FTP connections to node A will be successful again.

4. Conclusion

To the best of our knowledge, this article presents the first real-life implementation of a cooperation enforcement mechanism based on reputation which does not have an impact on underlying routing protocol adopted in the MANET (as opposed to [10]). Cooperation enforcement represents a fundamental building block for a heterogeneous MANET where no a priori trust relationships can be established among peers. The implementation of the software component described in this work constitutes a starting point for a thorough analysis (through measurements) of the impact of a cooperation enforcement mechanism on the operation of a heterogeneous MANET. For our future work we plan to investigate the behavior of the system in heavy load conditions to test the effectiveness of promiscuous listening. We will work on the fine tuning of the system parameters illustrated in (1) and study the introduction of an adaptive observation sampling factor to mitigate the CPU load generated by promiscuous listening and improve the responsiveness of the system behavior to the network conditions. We plan as well to investigate the impact of different reputation functions on the system accuracy.

References

[1] http://www.ebay.com

[2] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica, Free-Riding and Whitewashing in Peer-to-Peer Systems, ACM SIGCOMM'04 Workshop on Practice and Theory of Incentives in Networked Systems (PINS), August 2004

[3] Kevin Lai, Michal Feldman, Ion Stoica, John Chuang, Incentives for Cooperation in Peer-to-Peer Networks, in Proceedings of Workshop on Economics of Peer-to-peer Systems, June 5-6 2003

[4] Pietro Michiardi, Refik Molva, Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks, in Proceedings of European Wireless 2002 Conference

[5] Pietro Michiardi, Refik Molva, CORE: A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks, IFIP CMS02, Communication and Multimedia Security Conference, September 2002

[6] Pietro Michiardi, Cooperation enforcement and network security mechanisms for mobile ad hoc networks, PhD Thesis

[7] Sergio Marti, T.J. Giuli, Kevin Lai, Mary Baker, Mitigating routing misbehavior in mobile ad hoc networks, MobiCom00, International Conference on Mobile Computing and Networking, 2000

[8] http://www.tcpdump.org

[9] http://www.netfilter.org

[10] Sonja Buchegger, Jean-Yves Le Boudec, Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes Fairness In Dynamic Ad-hoc NeTworks, in Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), 2002

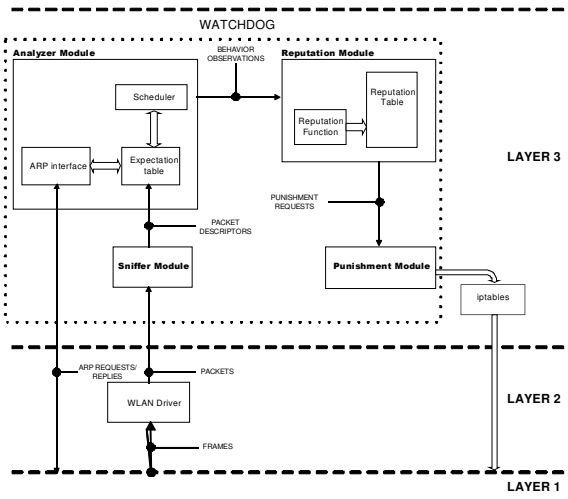


Figure 1. CORE Implementation Architecture

VoIP Testbed in Ad Hoc Networks

Jose Costa-Requena, Mohammand Ayyash, Jarrod Creado, Jarkko Hakkinen, Raimo Kantola and Nicklas Beijar
Networking Laboratory- Helsinki University of Technology
Otakaari 5, 02510 Espoo, P.O. Box 3000, Finland

***Abstract*— Ad hoc networks provide networking capabilites without infrastructure support. Ad Hoc networks have been researched for few years but still they do not get enough interest form the industry. The reason is that they do not fulfil any clear user needs. The actual wireless technologies already provide users with enough coverage and the necessary network capabilities for supporting high variety of peer to peer applications. In this paper we present VoIP application on top of Ad Hoc networks as a disruptive technology for voice communications without infrastructure support. With Ad hoc networks a low cost connectivity for VoIP, messaging and presence services can be provided and we present the results from a real VoIP testbed over Ad hoc networks.**

***Index terms*—Ad Hoc, VoIP, Service Discovery**

A. INTRODUCTION

Voice communications have been a disruptive technology in the past. Wireless telephony and Voice over IP (VoIP) can be considered as the next disruptions associated with voice communications. The next telephony innovation is linked to new drivers such as mobility and cost. In this paper we present the VoIP over Ad hoc networks as the next disruption related to voice communications. The feasibility of VoIP over Ad hoc networks is demonstrated with real testbed results. The results show the possibility of implementing voice communications with a reasonable quality of service (QoS). In order to implement the VoIP service we need to glue together Ad hoc technology with the right signalling and transport protocol.

B. AD HOC NETWORKING

The strength of an Ad Hoc network resides in the growth of IP over wireless and the self-organized networking feature that will enable pervasive and ubiquitous computing. Ad Hoc networks are seen as a suitable technology for embedded network devices in multiple environments such as vehicles, sensors, mobile telephones and personal appliances.

Ad Hoc networks present some new and unusual challenges that had not been primary concerns in fixed network deployment. Ad hoc devices should perform their own network topology functions keeping track of the connection between nodes and performing routing functionality. The link state in an Ad Hoc network changes whenever the users move, and the nodes must be able to provide automatic topology establishment and maintenance. Ad Hoc nodes need to be self-contained and have their own device discovery and control paradigms. The nodes need a set of mechanisms to allow a device to be automatically integrated and configured as part of an Ad Hoc network.

In Ad Hoc networks a new paradigm based on on-demand routing, where the nodes search for the routes only when needed, suits better than the existing fixed routing mechanism, where the nodes maintain the complete network topology. However, this routing does not scale for large networks. Therefore, numerous routing protocols and algorithms have been proposed. However, the absence of performance data in non-trivial network configurations continues to be a major problem. The Ad hoc routing protocols are typically subdivided into two main categories: proactive routing protocols and reactive on-demand routing protocols.

Reactive on demand routing protocols such as AODV [1], establish the route to a destination only when demanded. Proactive routing protocols such as OLSR [2] are derived from legacy Internet distance-vector and link-state protocols. They attempt to maintain consistent and updated routing information for every pair of network nodes by propagating route updates.

C. VOIP SERVICE IN AD HOC NETWORKS

At the early days of mobile communications, operators did not see relevant investing in mobile technologies. Today VoIP is already the next disruption in voice communications. Users are driven by cost and they start using Voice over IP applications that provide cheap voice communications. VoIP over Ad hoc networks will be the next distruptive technology. However, the voice quality is poor and users may not allow other people traffic to go through their devices.

1st. VoIP architecture

VoIP services require a signalling protocol that supports the routing and addressing. There are several protocols used for initiating and controlling multimedia sessions and in our testbed we have selected the Session Initiation Protocol [3]. The media transport is exchanged using the Real Time Transport protocol RTP [4].

A SIP session usually involves a User Agent (UA), a Proxy Server, a Registrar Server, and a Location server. Figure 1 represents a SIP registration flow diagram. The session set up is initiated through the SIP Registrar that can locate the destination SIP UA and acts as a proxy.

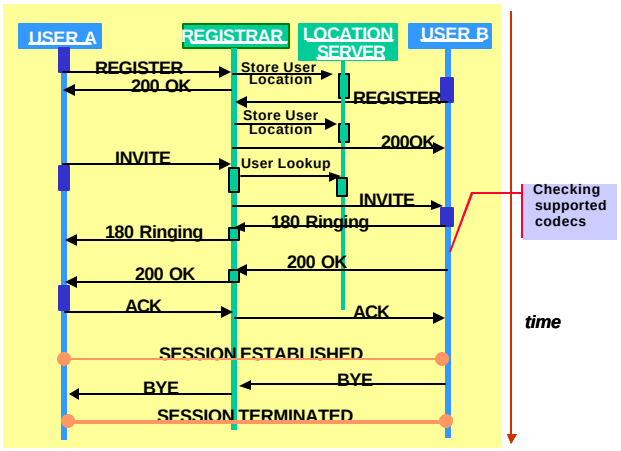


Figure 1. SIP UA Registration and session setup.

Moreover, a SIP UA can obtain the destination IP address using embedded service discovery mechanism without network support. Figure 2 presents the flow diagram for initiating a session in this case.

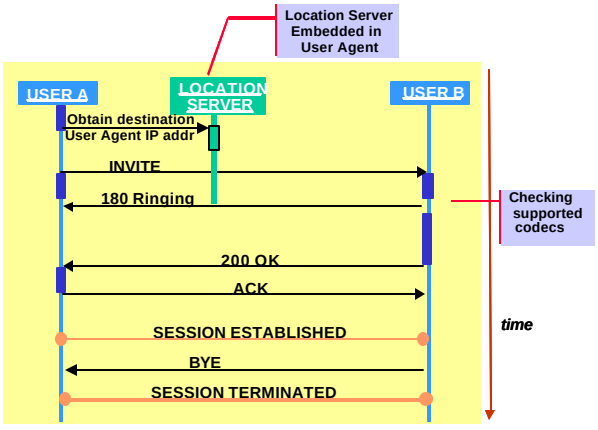


Figure 2. Session setup without network support.

D. VoIP Test bed

The VoIP testbed we have developed uses components from other research institutions [5]. In order to increase the QoS in Ad hoc networks, enhancements in the VoIP application have been implemented. The experiment was carried out in the laboratories of the Consiglio Nazionale delle Ricerche (IIT-CNR), Pisa, Italy. The test logs and additional information can be found in the project site at networking laboratory [6]. The experiment measures the overall performance of audio sessions using VoIP in Ad hoc wireless LAN environment.

The VoIP client is runing in both Laptops and iPAQs and additionally they require the following software: a GSM library [7] and an RTP library [8]. The tests are performed with two different routing protocols; OLSR and ADOV. We analyse the routing protocol effect in the overall performance.

Jitter buffer length is adjusted to analyse the effect in the quality of the audio session. Increasing it will reduce the perceived pauses in audio playback, resulting in smooth playback. On the other hand, this will increase the overall delay. Different RTP payload is applied per message, which is equivalent to increasing the amount of audio data in each RTP packet. This will enhance the audio playback at the receiver since each packet holds enough audio data to play until the next packet arrives. In addition, if one packet is lost, a larger mount of audio data is lost resulting in longer pauses.

2nd. Test metrics

Table 1 shows the metrics used to analyse the performance.

Table 1. Performance test metrics

Metrics	
RTP Traffic	Ratio between total RTP bytes sent divided by the total bytes sniffed at the transmitter. This represents the RTP traffic percentage of the total channel capacity.
GSM data	Ratio between total GSM bytes sent divided by the total bytes sniffed by Ethereal tool. This represents GSM traffic percentage of the total channel capacity. GSM bytes are calculated by multiplying the total number of RTP packets by the number of GSM packets per RTP packet, and multiplied by the GSM packet length, which is 33 bytes.
Signalling Overhead	Ratio between total number of routing protocol sent and received messages divided by the total number of GSM bytes successfully sent.
Overall QoS	Ratio of RTP packets successfully received out of

	those intended to be transmitted.
Loss in Link	Ratio of RTP packets lost in the link divided by the RTP packets intended to be transmitted.
Jitter	Jitter measured at the receiver during a continuous reception time.

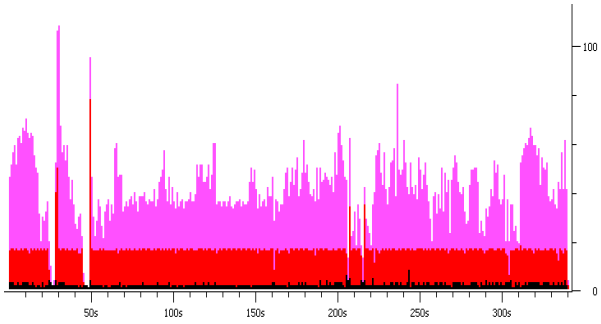
3rd. Test results

Table 2 presents results using OLSR and AODV routing protocols with 3 GSM packets per RTP message, where a 60ms and a 100ms GSM buffers are compared.

Table 2. OLSR and AODV results

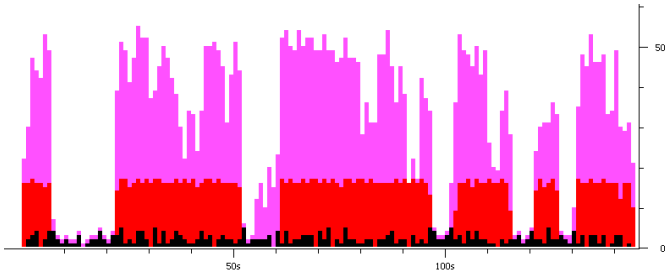
	OLSR		AODV	
Metrics	**60ms**	**100ms**	**60ms**	**100ms**
RTP Traffic	40.79%	40.79%	39.46%	39.46%
GSM data	26.40%	26.40%	25.47%	25.47%
Signaling Overhead	8.95%	8.95%	12.64%	12.64%
Overall QoS	60.11%.	53.57%	93.88%	77.03%
Link Loss	39.89%.	46.43%	6.12%	22.97%
Jitter	0.527078	0.045297	0.073521	0.0555660 2

Figure 3 and Figure 4 show the percentage of signalling traffic versus voice data.



Legend: Non RTP Traffic, RTP Traffic, AODV Traffic

Figure 3. AODV signalling overhead



Legend: Non RTP Traffic, RTP Traffic, OLSR Traffic

Figure 4. OLSR signalling overhead

4th. Test conclusions

The results show that when sending the same percentage of RTP and GSM packets the signalling overhead is bigger for OLSR (i.e. 3%). The OLSR has to send periodically link state updates.

Increasing the GSM buffer in OLSR helps to decrease the jitter. In case of a link broken AODV reacts immediately and re-establishes the link. Thus, the buffer helps to maintain a constant flow of audio packets. However, in case of OLSR when the link is broken the packet loss increases and the buffer has to store the received packets until the link is re-estabilised.

E. CONCLUSIONS AND FUTURE WORK

This paper presents a real VoIP testbed in Ad hoc networks. The architecture proposes an integrated routing functionality together with QoS optimisations for voice sessions in the application layer. The work under development consists of dynamically changing the number of audio packets per RTP message. Therefore, when the packet loss increases, the number of GSM packets is reduced within each RTP message in order to reduce the overall audio loss. Thus, when the link is stable and the packet loss is lower, the number of GSM packets increases in order to increment the audio packets that reach the destination.

Acknowledgment

We would like to thank the MobileMAN members who contributed to the testing in Pisa.

This work was partially funded by the Information Society Technologies (IST) program of the European Commission, Future and Emerging Technologies under the IST 2001-38113 MOBILEMAN project.

This work has been partly supported by the European Union under the E-Next Project FP6-506869

References

[1] C. E. Perkins and E.M. Royer, "Ad-hoc On Demand Distance Vector Routing", Second IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100, February 1999.

[2] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Lanouiti, L. Viennot and T. Clausen, "draft-ietf-MANET-olsr-02.txt", July 2000.

[3] M. Handley, H. Schulzrinne, E Schrooler, J. Rosenberg, et. al.,. "Session Initiation Protocol", RFC 3261, IETF.

[4] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889.

[5] AODV-UU, AODV implementation created at Uppsala Universtity (http://user.it.uu.se/~henrikl/aodv/).

[6] Helsinki University of Technology, Networking Laboratory (http://www.netlab.hut.fi/~mayyash/pisa/index.htm).

[7] GSM library v06.10 (http://kbs.cs.tu-berlin.de/~jutta/toast.html)

[8] RTP library v2.9 (http://research.edm.luc.ac.be/jori/jrtplib/)

Experimenting a Layer 2-based Approach to Internet Connectivity for Ad Hoc Networks*

R. Bruno, M. Conti, E. Gregori, A. Pinizzotto
IIT Institute
National Research Council (CNR)
Via G. Moruzzi, 1 - 56124 PISA, Italy
Email: {*r.bruno,m.conti,e.gregori,a.pinizzotto*}@iit.cnr.it

E. Ancillotti
Dept. of Information Engineering
University of Pisa
Via Diotisalvi 2 - 56122 Pisa, Italy
Email: emilio.ancillotti@iet.unipi.it

Abstract

A prerequisite for the mass-market deployment of multi-hop ad hoc technologies is the capability of integrating with existing wired infrastructure networks. However, current solutions to support connectivity between ad hoc networks and the Internet are based on complex mechanisms, such as Mobile-IP and IP tunnelling. In this paper we propose a lightweight solution based on simple Layer-2 mechanisms. Experiments carried out in a real test-bed confirm the validity and efficiency of our approach.

1. Introduction

In spite of the massive efforts in researching and developing mobile ad hoc networks in the last decade, this type of networks has not yet witnessed a widespread deployment. The low commercial penetration of products based on ad hoc networking technologies could be explained by considering that users are interested in general-purpose applications where high bandwidth and open access to the Internet are consolidated and cheap commodities. For these reasons, there is a growing interest in designing working mechanisms for providing an easy access to the Internet to nodes in ad hoc networks. In this paper, we address this issue proposing a novel approach to ensure a working Internet connectivity to proactive ad hoc networks. We decided to consider proactive routing protocols because this family of routing protocols is more suitable than reactive protocols for supporting advanced services and applications in mobile ad hoc networks.

Two classes of approaches have been proposed so far to support connectivity between ad hoc networks and the Internet. One approach is to implement a Mobile IP Foreign Agent (MIP-FA) in the ad hoc node that acts as Internet gateway, and to run Mobile IP in all the ad hoc nodes [1]. A different approach relies on the implementation of a Network Address Translation (NAT) in the gateway [3], that translates the IP addresses of ad hoc nodes to an address on the NAT gateway, which is routable on the external network. Such approaches are based on complex IP-based mechanisms originally defined for the wired Internet, like the IP-in-IP encapsulation, Mobile IP and explicit tunnelling, which may introduce significant overheads. This paper proposes a *lightweight* technique to provide global Internet connectivity to the ad hoc nodes, using only Layer-2 mechanisms. The basic idea is to logically extend a wired LAN to the ad hoc nodes in a transparent way for the wired nodes by developing a specific *Proxy ARP* daemon inside the gateway. Experiments carried out in a real test-bed confirm the validity and efficiency of our approach.

2. Background on the OSLR Protocol

Our test-bed uses the OLSR protocol as ad hoc network routing algorithm [2], although our solution can be applied to any proactive scheme. The OLSR algorithm employs an efficient dissemination of the network topology information by selecting special nodes, the multipoint relays (MPRs), to forward broadcast messages during the flooding process. In order to allow the injection of external routing information into the ad hoc network, the OLSR protocol defines the Host and Network Association (HNA) message. The HNA message binds a set of network prefixes to the IP address of the node attached to the external networks, i.e., the gateway node. In this way, each ad hoc node is informed of the network address and netmask of a network that is reachable through each gateway. Hence, OLSR exploits the mechanism of *default routes* to advertise Internet connectivity. For instance, a gateway that advertises the $0.0.0.0/0$ default route, will receive all the packets destined to IP addresses without a route on the local ad hoc network.

*This work was funded by the Italian Ministry for Education and Scientific Research (MIUR) in the framework of the FIRB-VICOM project, and by the Information Society Technologies program of the European Commission under the IST-2001-38113 MobileMAN project.

3. Internet Connectivity using a *Proxy ARP*

The proposed architecture is depicted in Figure 1. An OLSR-based ad hoc network is interconnected via a *Master Gateway* (*MG*) to a wired LAN, which provides the connectivity to the external Internet. The wired LAN is an IP subnetwork identified by $\overline{IP}_S/L$, i.e., an IP network address ($\overline{IP}_S$) and the network mask length L (for example $X.Y.96.0/22$). The wired nodes' IP addresses belong to $\overline{IP}_S/L$. Thus, the wired hosts are able to exchange packets using their *ARP table*, which is a list of mappings between the IP address (Layer-3) and the MAC address (Layer-2) of known hosts on the same subnet. The connectivity between the wired LAN and the Internet is provided by a router and standard IP routing protocols.

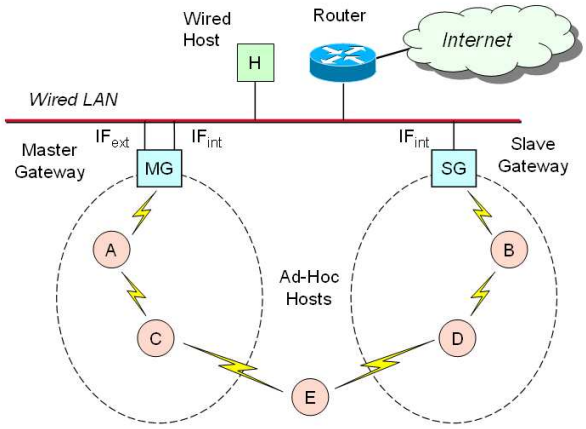


Figure 1. Test-bed implementation.

Our OLSR-based ad hoc network consist of: (i) mobile ad hoc nodes and (ii) a fixed backbone formed by one *MG* node and multiple *Slave Gateways* (*SGs*), connected via wired links. The *MG* is a node with two wired interfaces and a single wireless interface that acts as gateway allowing the ad hoc network to be connected to the Internet. The *SG* nodes have two interfaces, wireless and wired, which ensures the routing between separated parts of the ad hoc network through the wired infrastructure. The ad hoc routing protocol is implemented by an *OLSRd* daemon, which runs in all the interfaces - wireless and wired - of mobile nodes, *SG* nodes and *MG* node, with the exception of the *MG* node's wired interface IF_{ext}, which provides access to the external network, advertising Internet connectivity as default routes (i.e., through HNA messages). Mobile nodes are configured with a static IP address belonging to the same subnet of the wired LAN, i.e.,$\overline{IP}_S/L$. The hybrid nodes' interfaces are configured with private IP addresses. This is done to facilitate the configuration of the *MG* node, as explained later. It is worth remarking that we devised the *SGs* nodes as simple entities supporting the ad hoc node mobility, and increasing the available bandwidth between the *MG* node and the mobile nodes. For this reason they do not need a globally routable IP address.

Connectivity for Outgoing Traffic. The *OLSRd* daemon builds the routing tables with entries that specify the IP address of the next hop neighbour to contact to send a packet destined to another host or subnetwork. Since the *MG* node advertises $0.0.0.0/0$ as default route, all packets destined for IP addresses without a route on the ad hoc network, will be routed along the default route to the *MG* node and forwarded to the Internet. A special case is when a mobile node wants to send a packet addressed to the node on the local wired LAN (e.g., node *H* in Figure 1). As the destination IP address, IP_H, belongs to $\overline{IP}_S/L$, the routing table lookup on the source node will assume that the destination node is directly connected to the node wireless interface. This will result in a failed ARP Request for the IP_H address, sent on the source node's wireless interface. To solve this problem, we split the original $\overline{IP}_S/L$ subnet into two consecutive subnets, $\overline{IP}_{SL}/(L+1)$ and $\overline{IP}_{SU}/(L+1)$, such as to have $\overline{IP}_S/L = \overline{IP}_{SL}/(L+1) \cup \overline{IP}_{SU}/(L+1)$. For example, $X.Y.96.0/22 = X.Y.96.0/23 \cup X.Y.98.0/23$. Thus, the *MG* node has to advertise also these two subnets on HNA messages. In this way, each mobile node will always have, for any host *H* on the local wired LAN, a routing table entry with a more specific network/mask than the one related to its wireless interface (i.e., $\overline{IP}_S/L$). Consequently, the longest-match criterion, in the routing table lookup, will determine the right next hop for the IP_H address.

Connectivity for Incoming Traffic. To allow the nodes on the local wired LAN, including the router, to send packets to the mobile nodes, a specific daemon runs on the IF_{ext} interface of the *MG* node, called *Ad Hoc Proxy ARP daemon* (*AHPAd*). This daemon periodically checks the Master Gateway's routing table and ARP table, such as to *publish* the *MG* node's MAC address for each IP address having an entry in the routing table with a netmask $255.255.255.255$. The entries related to the IF_{int} interface of the *MG* node and the *SG* nodes' interfaces are excluded. Hence, the *MG* node acts as a Proxy ARP for the mobile nodes. When an host on the local wired LAN has to send a packet to a mobile node in the ad hoc network, it deems the destination on its wired network. Then, the node checks its ARP table for a IP-MAC mapping and, if it is not present it sends an ARP Request. The *MG* node answers with an ARP reply providing its MAC address and the packets can be correctly sent to the destination through the *MG* node.

3.1. Experimental Results

In our test-bed we used the OLSR_UniK implementation for Linux in version $0.4.8$, adopting the default setting for each protocol parameter. All nodes where located in the same room, and the *iptables* feature of Linux was used to emulate that all nodes were not in radio visibility of each other. To

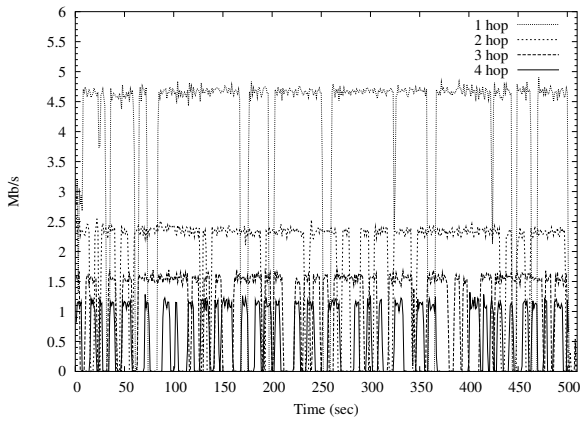


Figure 2. Throughput of a single TCP flow for different chain length.

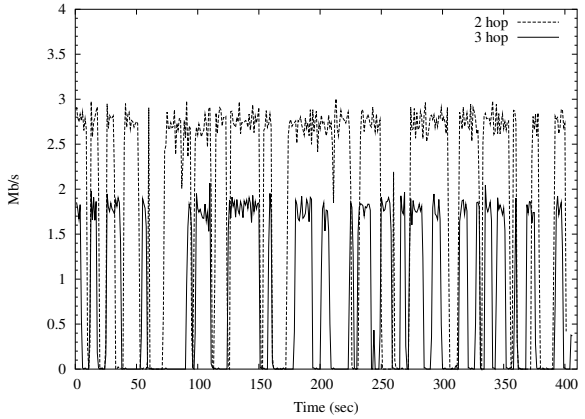


Figure 3. Throughput of a single TCP flow with mobility.

generate the asymptotic TCP traffic during the experiments we used the *iperf* tool.

We performed a first set of experiments aimed at evaluating the impact on the TCP throughput of the number of wireless hops traversed in the ad hoc network to reach the *MG* node. The experimental results are shown in Figure 2. As expected, the longer the route, the lower is the throughput achieved by the TCP flow, and the throughput decrease follows an almost linear relationship. The figure shows also that, although the nodes are static, the TCP throughput is not stable, but the TCP flow could be in a stalled condition for several seconds. This can be explained considering that the control frames are broadcast frames that are not acknowledged, hence more vulnerable to collisions and channel errors than unicast frames. As a consequence, losses of control frames can induce the loss of valid routes.

The second set of experiments was carried out to verify the impact of mobility on TCP throughput. We allowed node E (C) to alternate between being within radio range of C (A) and being within radio range of D (B), on 50 seconds time interval (see Figure 1). To emulate a *soft* handoff, we also allowed the mobile node E (C) to have both C and D (A and B) within radio range for 10 secons at the beginning of each interval. The experimental results are shown in Figure 3. The curve with label "*3 hops*" refers to the node E's mobility, while the curve with label "*2 hops*" refers to the node C's mobility. The figure shows that the maximum throughput the TCP connection achieves, is not affected by the mobility. This is due to the fact that our solution doesn't need any IP encapsulation to work, differently from [3]. However, the TCP instability increases because the OLSR neighbour sensing mechanism may take up to 6 seconds to discover the link change.

4. Concluding Remarks

To conclude this paper we discuss on limitations of the proposed approach and on the enhancements currently under investigation.

- On the *MG* node two wired interfaces are needed because the Proxy ARP does not allow to answer to ARP Requests for IP addresses that are reachable through the same interface on which the ARP Request was received. Nevertheless, they can be replaced by a single wired interface and emulated by a bridging function and two virtual interfaces.
- The proposed architecture is working only with proactive ad hoc routing protocols. In fact, the Proxy ARP function on the *MG* node needs to know all the mobile nodes' IP addresses, which has to publish on its ARP table.
- Only one *MG* node, i.e., access point to Internet, is allowed in the ad hoc network to avoid conflicts between different Proxy ARPs. The redirect mechanism could be exploited to support multi-homed (i.e., with multiple *MG* nodes) ad hoc networks.
- The IP addressing inside the ad hoc network is static. Address auto-configuration techniques should be integrated in our architecture.

References

[1] M. Benzaid, P. Minet, K. Al Agha, C. Adjih, and G. Allard. Integration of Mobile-IP and OLSR for a Universal Mobility. *Wireless Networks*, 10(4):377–388, July 2004.

[2] T. Clausen and P. Jaquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, October 2003.

[3] P. Engelstad, A. Tønnesen, A. Hafslund, and G. Egeland. Internet Connectivity for Multi-Homed Proactive Ad Hoc Networks. In *Proc. of ICC'2004*, volume 7, pages 4050–4056, Paris, France, June 20–24 2004.

Demo of Ana4: an Hybrid Local Area Ad hoc Network Architecture

Nicolas BOULICAULT, Guillaume CHELIUS and Eric FLEURY
CITI – INRIA/ARES
INSA de Lyon – France

Abstract

We present the implementation of Ana4, a practical architecture suitable for interconnecting devices in an as hoc hybrid network environment, *where wired and multi hop wireless technologies are used. Ana4 is a 2.5 ad hoc layer that allows a full compatibility with TCP/IP.*

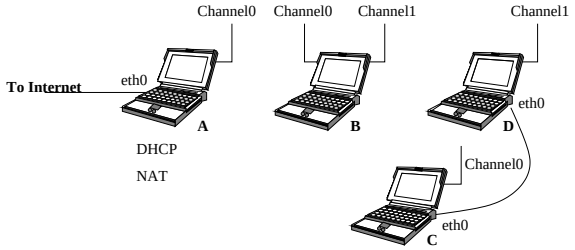


Figure 1. Ana4 test bed proposal.

1 Project description

In order to extend the coverage of actual wireless devices, wireless ad hoc or mesh networks have been proposed. The aim of this demo is to set up a small hybrid mesh network based on Ana4 [2, 3, 4, 5, 1], an interconnection architecture suitable for heterogeneous and spontaneous networks that mix various kinds of link layer technologies: wired and wireless multi hops. By ad hoc architecture, we denote a set of rules and operations dealing with addressing and routing that must be set up for the ad hoc network to offer basic services such as ad hoc connectivity, TCP/IP compatibility, Internet connectivity and vertical handoff support. During this demo, we propose to set up an hybrid ad hoc test bed based on Ana4, mixing wire and wireless media and offering a full auto-configuration support, a full TCP/IP compatibility and connection to the Internet

The Figure 1 illustrates the problem of interconnecting heterogeneous devices all together in an hybrid ad hoc network. This topology will be used as the reference during the demo. Host A plays both the role of gateway through its wire interface `eth0` connected to the Internet and the role of "base station" via its wireless interface `wlan0` on a given channel 0. Host A also runs a DHCP server and performs IP masquerading to connect the Internet to the ad hoc network. In node A, only the wireless interface `wlan0` is part of the ad hoc network and configured in Ana4 ad hoc mode. Host B has two wireless interfaces, one on channel 0 and one on channel 1. Note that the two link layer interfaces (`wlan0` and `wlan1`) participate to the ad hoc network and are gathered under the same Ana4 virtual interface `adh0`. Hosts C and D both have one wireless interface `wlan0` and one Ethernet interface `eth0`. For nodes C and D both wireless and wire interfaces participate to the ad hoc network and are gathered under one Ana4 interface `adh0`.

The Figure 2 illustrates the layers and abstractions that we encounter in a generic Ana4 network. In our test bed illustrated in Figure 1 we have at the physical layer three separate physical connectivity graphs corresponding to channel 0 and channel 1 of the 802.11 link layer and to the Ethernet link. At the ad hoc layer we merge this three graphs on an "ad hoc node" basis. That is, a node is no more a link layer 2 interface but it is the gathering of all interfaces that participate to the ad hoc network (all wireless interfaces/channels and Ethernet cards in our case). At the IP layer, the ad hoc network is seen as an Ethernet link.

The aim of this demo is to demonstrate the advantages of Ana4 and its compatibility with classical uses

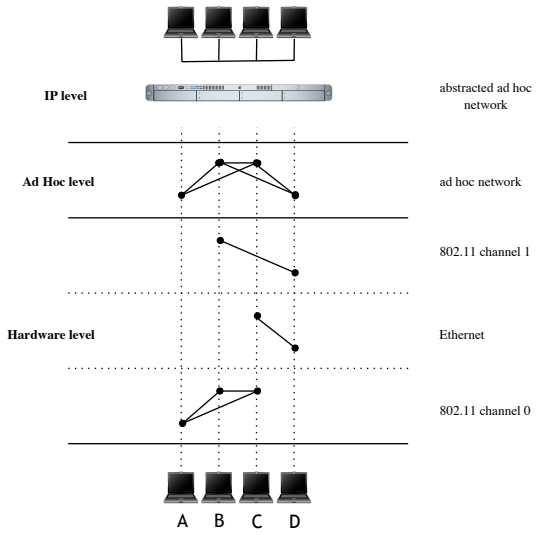


Figure 2. The different abstractions in an Ana4 network

of TCP/IP. For example, When host *D* joins the network, it uses a classical DHCP client to broadcast a request on the local network in order to retrieve its IP address, its DNS and its gateway. Despite the fact that this local network is a multi- hop, multi-interface network, the broadcast packet is flooded inside the mesh network thanks to Ana4 and the response is forwarded hop by hop back to node *D*.

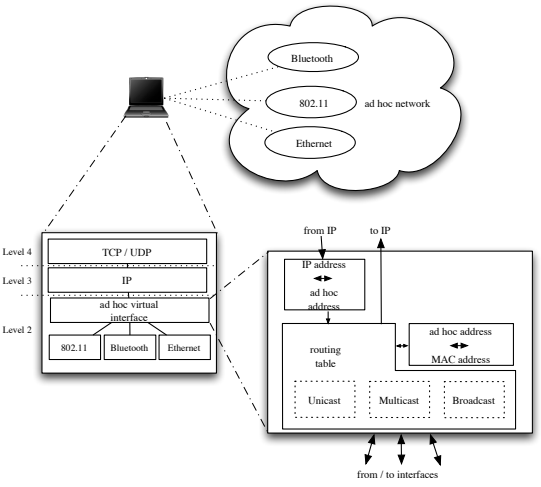


Figure 3. The ad hoc virtual interface

The Ana4 architecture is based on the notion of virtual ad hoc interface, illustrated in figure 3. The simple but efficient principle behind the virtual interface is to hide the different physical devices and hardware networks behind the illusion of a single virtual network. At the ad hoc level, this virtual network is a wireless multi-hop network; at the IP level, it is a switched Ethernet link. Another powerful characteristic of this architecture is to allow an host to use a device simultaneously in ad hoc and in classical modes. Suppose that a physical device handled by a virtual ad hoc interface is also configured as an Internet device. From the IP view, the mobile hosts two distinct interfaces. IP networking is performed over these two interfaces without interference. This will also be shown during the demo.

References

[1] N. Boulicault, G. Chelius, and E. Fleury. Experiments of ana4: An implementation of a 2.5 framework for deploying real multi-hop ad hoc and mesh networks. In *IEEE ICPS Workshop on Multi-hop Ad Hoc Networks: from theory to reality (REALMAN 05)*, Santorini, Greece, July 2005.

[2] G. Chelius and E. Fleury. Ananas : A local area ad hoc network architectural scheme. In *MWCN 2002*, Stockholm, Sweden, Sept 2002. IEEE.

[3] G. Chelius and E. Fleury. Ananas : A new adhoc network architectural scheme. RR 4354, INRIA, 2002.

[4] G. Chelius and E. Fleury. Design of an hybrid routing architecture. In *Fifth IEEE International Conference on Mobile and Wireless Communications Networks (MWCN 2003)*, Singapor, October 2003. IEEE, IFIP.

[5] G. Chelius and E. Fleury. Request for an ad hoc addressing architecture. In *Wireless Personal Multimedia Communications (WPNC)*, Abano Terme, Italy, September 2004.

Haggle Architecture and Demo of its Real World Implementations

Pan Hui[*], Jon Crowcroft[*], James Scott[#], Christophe Diot[#] ,Augustin Chaintreau[#] ,Richard Gass[#]

[*]*University of Cambridge*
firstname.lastname@cl.cam.ac.uk

[#]*Intel Research Cambridge*
{ james.w.scott, christophe.diot, augustin.chaintreau, richard.gass}@intel.com

Abstract

Pocket Switched Networks (PSN) are a new communication model at the intersection of Mobile Ad-hoc Networks (MANET) and Delay Tolerant Networks (DTN).We propose a general software architecture, the Haggle Architecture for PSN, and demonstrate it using a prototype including two PSN applications, namely distributed file sharing and newsgroups.

1. Introduction

Mobile Ad-hoc Networks (MANET) research often assumes network conditions including long-lived paths and full connectivity, and is therefore targeted at situations where the network is dense and static. Previous work in Delay Tolerant Networks (DTN) [1] has focused on partitioned networks with occasional links between them, e.g. due to satellites or "message ferries" [2]. Both models are not realistic when considering our target application scenario, namely for humans moving around during their daily lives. We observe that, while humans often have end-to-end connected paths via "islands" of Internet connectivity, this is not always true, e.g. while traveling. It is also unrealistic to assume that mobile humans will have a wireless network path with other humans which they wish to perform networking with. We proposed Pocket Switched Networks (PSN) as a more realistic model for human daily life. PSN is designed around real life human mobility measurements [3][4] and using a top-down approach that is a practical and realistic communication model based on what kind of applications human need or may like to have in their daily lives. PSN realizes the presences of gaps between the "islands of connectivity" and makes use of local connectivity and user mobility during these gaps, and global connectivity for data delivery when it is available.

Haggle is a general architecture proposed for PSN applications. It embraces "Application Layer Framing" [5], with application-layer and network-layer information collapsed into on space. It use Application-level Data Units (ADU) as basic unit of communication and use attribute-value tuples to identify communication parties and different applications.

A distributed file sharing system and a newsgroup making use of only human mobility and local connectivity are implemented using the Haggle Architecture. The implementation uses Java J9 platform running on PDA and PC and Bluetooth and WiFi as underlying network connecting media. During the demo, the Bluetooth version will be demonstrated.

2. Haggle Architecture

We now introduce Haggle, an implementation of Pocket Switching which we are using to prototype solutions to the challenges identified in Section 3. We expect that the full specification and implementation of Haggle will take several years. Development will involve a process of iterative refinement, driven by feedback from the deployment of trial applications. We describe below the architectural elements that are present in the initial prototype of Haggle.

2.1. Application-level Data Unit

In the spirit of "Application Layer Framing" proposed by Clark and Tennenhouse in the early nineties [5], the Haggle architecture is not layered; application-layer and network-layer information are collapsed into one space. We represent messages as application-level data units (ADUs), which are comprised of a number of attribute-value pairs.

A node decides whether to offer an ADUs to a neighbour based on many attribute-value pairs, rather than simply on the destination address as in IP-centric networking. This is useful because, *in Haggle, an ADU's destination can be loosely specified, or not specified at all*. A sender may not know the set of devices available to the recipient. Alternatively, for message types such as information queries, many devices could take the role of destination.

A node might accept transmission of an ADU from a neighbour for two reasons. One is that the node may determine that it is a valid recipient, since it might carry an application which is interested in the ADU. The other is that the node might decide that it has a high expectation of further forwarding opportunities for that ADU, for example because the recipient was recently seen by it, or because the node expects to acquire global connectivity shortly, and can thereby deliver the message. Again, the visibility of application-level information in various attribute-value pairs may usefully contribute to a node's decision.

Examples of possible attribute-value pairs for various Haggle applications are:

```
message-type: private message
originator: NAME James Scott
sender-authentication: [cryptographic signature]
recipient: NAME Pan Hui
recipient: SMTP pan.hui@cl.cam.ac.uk
recipient: SMS +441233764432
encrypted-data: [encrypted message]

message-type: query
data: opportunistic networking

message-type: public message
forum: recipes
subject: chicken madras
```

2.2. Node Architecture

The preliminary design of the Haggle node architecture, illustrated in Figure 1, is based around the notion of a *Haggle Information Store* (HIS) containing ADUs on each user device. Applications can inject new ADUs into the HIS, can register interest in incoming ADUs by specifying matching criteria for particular attribute-value pairs, and can search the HIS for existing ADUs.

The *Haggle control* module decides whether to accept an incoming ADU into the HIS, provides an API for applications, and controls the forwarding module. The *forwarding* module performs neighbour discovery using the available network interfaces, and when necessary connects to neighbours which are identified as potential nexthops for ADUs in order to offer those ADUs for transmission. Note that, where it is available, the Internet is regarded as just one of many network interface types.

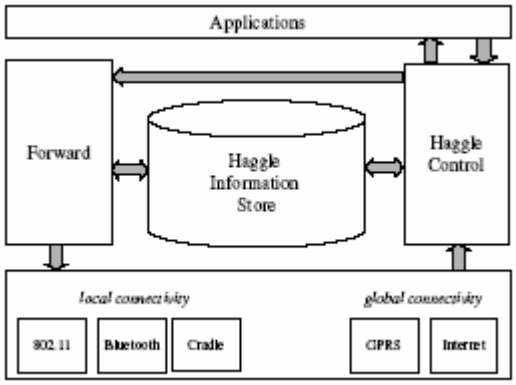


Figure 1: Prototype Haggle Architecture

2.3.Implementing Haggle

We aim to develop Haggle for *mobile computing devices*, which obviously includes notebook PCs and PDAs. Recently, this term has also become applicable to mobile phones, which now have significant storage, computing power, local networking (generally in the form of Bluetooth), and support for dynamically-loaded applications.

We use Java in order to support a wide range of platforms with a single reference codebase. For local connectivity, we utilise Bluetooth and 802.11 in ad hoc mode. Conveniently, Bluetooth is available on all our targeted platforms. An initial step in the Haggle project has been to develop implementations of the Java Bluetooth standard (JSR-82 [6]) for Windows XP and Windows CE, which we have released under the LGPL open source license[3].

3. Haggle Demo

The demos we given here are the ad-hoc file sharing system and the ad-hoc newsgroup application. Both applications are implemented based on the Haggle architecture defined above. The implementation is done on Java J9 platform on PDA with Bluetooth as the transmission medium. Inside the demo, several PDAs running the Haggle applications will be distributed to the participants for testing.

3.1. File Sharing System

By using the Haggle File Sharing system, the users can do searching and downloading files in a totally ad hoc and opportunistic manner. Type a file name wanted to

search into the browser and a File Request ADU will be sent to everyone encountered If a device contained the file, it will generated a File Response ADU which will be forward out in two manners 1) send to the originator of the Request ADU when get into contact range, or 2) send controlled copies into the devices encountered which act as relays of this copy and would send to the originator of the request when encountered.

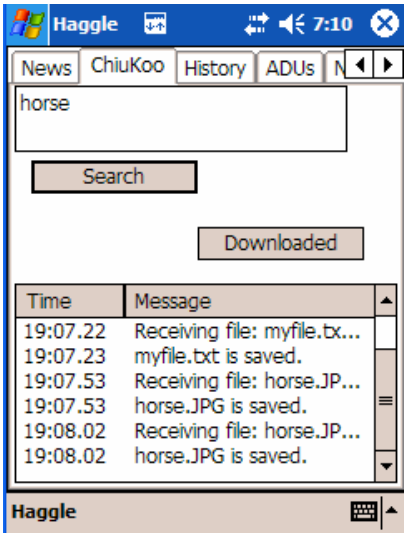


Figure 2: Haggle File Sharing User Interface

3.2. Newsgroup system

The Haggle Newsgroup allows the users to publish to an ad-hoc newsgroup. When two devices see each other, they will exchange ADU and the contents of the newsgroup will be updated.

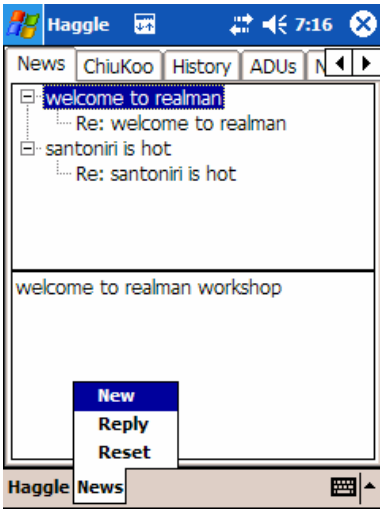


Figure 3: Haggle Newsgroup User Interface

4. Conclusions and Future Work

The Haggle Architecture for PSN is presented and two haggle applications, file sharing system and newsgroup are implemented on Java j9 platform running on PDA with Bluetooth. We will make the codes soon available under an open source license. The implementation of WiFi connectivity is proceeding.

Over the next few years, we aim to build more Haggle-based applications including instant messaging and web browsing with automatic use of neighbours' caches. We plan to test these applications by rolling out prototypes to local users and deploying them to larger user groups such as conference attendees. This will enable us to study aspects of Haggle including usability, scalability, network congestion, and user behaviour, which can only be conclusively studied in deployments.

5. References

[1] Kevin Fall. A delay-tolerant network architecture for challenged internets. In Proceedings of ACM SIGCOMM, 2003.

[2] Wenrui Zhao, Mostafa Ammar, Ellen Zegura, "A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks," Proceedings of ACM Mobihoc 2004, Tokyo Japan, May 2004.

[3] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. Technical Report UCAM-CL-TR-617, University of Cambridge, Computer Laboratory, February
2005.

[4] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, C. Diot, Pocket Switched Networks and the Consequences of Human Mobility in Conference Environments t, To be presented at the Workshop on Delay Tolerant Networking during SIGCOMM 2005.

[5] D. Clark and D. Tennenhouse. Architectural considerations for a new generation of protocols. In ACM SIGCOMM, pages 200–208, Sept. 1990.

[6] http://jcp.org/en/jsr/detail?id=82.

Demo of residual bandwidth estimation in an 802.11 ad hoc network

Martin Nielsen
University of Oslo and UniK
Department of Informatics
Oslo, Norway
martinnn@unik.no

Abstract

Quality of service in wireless LANs differ from their wired counterpart in that variation in available resources is not only due to competition by other traffic, but also due to variation in link quality. Various proposals exist to remedy the effects of this dynamicity. Our aim is to implement a system that by offering more information about the current per peer link state and resource usage can improve performance not only by provisioning through reservation, but also in route calculation. To aid in achieving this, a closed network has been constructed in order to conduct live experiments where outer factors like interference and other competing stations are controlled.

1. Introduction

Though mobile ad hoc networks (MANETs) might arise anywhere at any time one would not only want to support reliable end-to-end communication, but also some kind of premium service. For some scenarios of use, like battlefield deployment, it is a must. This complicates matters and presents a need for some sort of management of available and consumed resources. This is where quality of service (QoS) comes to play.

To accommodate a premium service certain knowledge about the surroundings has to be evident both in term of available hosts and their capabilities. The effort on our part is four fold:

- Sensing and probing in order to collect data concerning the link state.
- Bandwidth estimation with metrics for transmission and reception rates.
- Quality of service by adaptation of local flow policy and providing better routing metrics. The ability of the network to smooth things out and pick up general throughput might be seen as a service improving the general quality perceived by applications.
- Controlled live experiments. Though all the pieces necessary for a live network are covered one would still wish to control certain factors inherent to shared mediums during development. These are the hidden and the exposed node problem, node separation and general interference from items like microwave ovens and other appliances operating in the industrial, scientific and medical (ISM) bands that jam the communication in the vicinity of a receiver.

The demo will be of the two first objectives listed above; an ad hoc network where bandwidth estimation is conducted in the driver where the passive estimation technique implemented by CRC in Canada [4] is running alongside our active probing implementation. This will be in combination with a walk through the inner workings of the implementation where we look at how hardware and driver constraints impose limitations not observed in the safe and predictable surroundings of NS-2 simulations.

The test bed will consist of three laptops with two of them conversing and one listening and probing in order to estimate residual channel capacity. In the following sections we will examine the components of the demo and the implemented driver closer.

2. The link layer

It was argued in [3] that the 802.11 medium access control protocol (MAC) is not the best suited for ensuring QoS. Because of the commercial proliferation of compliant hardware it has however become the de facto standard. Its usability can nevertheless be augmented by offering more and better information to higher protocol layers. In that spirit the following sections will discuss how to estimate residual bandwidth and link quality in order to provide some useful MAC metrics.

2.1. Estimating residual bandwidth

In contrary to wired networks, bandwidth estimation is not a straightforward operation in shared medium radio networks. Though much research has been done and a theoretic analysis of one hop bandwidth estimation is in place [2] it is however not a simple task when multi hop scenarios and real hardware have to be addressed.

There has always been a gap between the theoretical analysis models and simulation results on one side and what is possible with current commercial products on the other side. Although results acquired by the former are indeed useful, existing hardware and their drivers put constraints on their applicability. This can be exemplified by looking at how we can do bandwidth estimation on the offered devices. Two methods of doing exactly this, the active and the passive estimation techniques, are described in the following sections. It must be noted however that these are neither exclusive of each other nor the only proposals of how to accomplish estimation.

2.2. Active probing

The active probing in [6] was designed mostly for one hop networks with constant modulation during the estimation. Our effort has tried to augment that scheme to encompass multi hop scenarios and to allow for use of the modulation deemed best by the drivers own rate controller. To begin we will first describe how the original proposal worked and then describe the additional considerations which had to be done in order to accommodate our use.

In its infancy this method started out by sending out a chain of probe packets to assess the channel occupancy. It essentially measures the time of each sequence of RTS/CTS/DATA/ACK and attributes any delay in excess of an estimated value to be caused by deferring from the medium entailed by either physical or virtual carrier sensing. By uniformly distributing probes within every transmission interval one can equally represent the channel occupancy. Thereby it is possible to infer the channel state. The authors suggested occupying 30 to 40 percent of the channel with probes. The channel utilization is estimated by applying the following formula:

$$U = \frac{\sum_{i=1}^{i=K_c} T_m^i - \overline{T}_p K_c}{K \Delta T}$$

Where U is the utilization estimate, T_m is the measured MAC delay, $\overline{T}$ is the calculated average MAC delay when the channel is sensed idle, K_c is the number of probes delayed more than predicted, K is the total number of probes and ΔT is the interval between the probes. The residual bandwidth is estimated by $n_{res} = (1 - U)n_{max}$ where n_{max} is the theoretical maximum throughput that can be calculated for a specific frame size, which implies that residual bandwidth is dependent on frame size.

2.3. Adaptation of the active probing scheme

The active probing proposal does have some glitches. The ones inherent when moving from a simulation environment to real hardware will be dealt with first, leaving the comparative details when looking at it in conjunction with the passive proposal described later.

Starting of with the universality of the method it has to be mentioned that it has a strong coupling to the driver that cannot be solved in user space when working under Linux. The method must measure the time from passing the packet onto the device till the exact time the ACK returns. To the best of our knowledge only one current driver allows such tight coupling, specifically the Atheros chipset MADwifi driver [5]. However, this driver is not completely open since its radio can be tuned to frequencies outside the ISM band, the FCC and other similar authorities have pressed for keeping a part closed. This lowers accuracy but is still better than the jiffy accuracy provided by the common wlan-ng-prism2 header found in most drivers.[1]

In addition the formula had to be custom tailored to the multi rate probes that could be sent during measurement. We do this by sorting the probing data into intervals based on the rate used and calculating the mean value.

The limitations of the stock Linux kernel also put a constraint on accuracy. The standard timekeeping accuracy is set to one millisecond for timed events. An addition is being developed to improve this [1], but has yet to be implemented by us. This gives us a slight problem because we are not able to control the channel saturation to a tight specific level. This issue does not break the technique however and is mostly a problem when trying to minimize saturation problems. In addition we do not account for clock drift, but as the time measured is per probe and not for the entire train it should not be a source of error.

During the demo we will look closer at what actually happens when sending a probe. Because we can see how many retires, what kind of protection and the rate we can reason closer about the events, and look if all the excess delay can be attributed to cross traffic. This will illustrate better the design issues for the simple probes. To se if the algorithm responds to changes in traffic load two other machines will vary it so we can inspect the predicted residual bandwidth. Though the conference room will undoubtedly be full of other wireless signals we can still see if the values are correlated.

[1] The information gathered by setting the interface in promiscuous or rf-monitoring mode is time stamped, at least in the madwifi driver, with jiffy precision. This is one ms in the 2.6 linux kernel which does not give the precision needed.

2.4. Passive probing

In order to minimize the intrusion on the network one can try to estimate residual bandwidth by listening to incoming traffic and adding to that channel occupancy seized while sending. Because the driver does not grant access to the network allocation vector (NAV) it is not possible to directly estimate the residual bandwidth by simply monitoring the busy state given by the carrier sensing state machine. What we can do instead is to take all the frames going in and out of a node and the ones heard through promiscuous mode to estimate how much time this has taken during an interval. CRC in Canada has implemented this technique in the Atheros chipset driver. The metrics provided by the driver are per neighbour node to achieve greater granularity.

To be able to accurately depict the channel state one has to pay great attention not only to the 802.11 frame format but also to the sequence of frames. Additional consideration has to be taken because one has to include the mean back off times for RTS and data frames, and also non-standard behaviour by the driver. We will try to send frames of different sizes and with various types of protection to see if we notice the difference. This is also a useful technique to see if the driver complies with our requests as other types of experiments can get interesting results when based on flawed assumptions.

2.5. Complementary performance

The two techniques both have their limitations. The active probing saturates the channel from 30 to 40 percent and is not good when load is heavy. It could also be a problem if two nodes within range probe at the same time. But in contrast to the passive technique it can estimate the intersection of the bandwidth at the sender and the peer. The peer can be overloaded by traffic we cannot hear and probing is a way to estimate this. As mentioned we do not have access to the NAV so it is not possible to estimate how often the physical carrier sensing is busy. Active probing defers from the channel and takes this into account.

We propose combining the two methods to harvest their benefits and reduce their negative sides. If the channel saturation is sensed above 50 percent one could argue that probing would be unwise and if no good information is available the active method is the way to go.

Since the nodes are to close to each other during the demo we will not be able to simulate a node chain showing how the two methods can complement each other. But we can run them in parallel and compare their results. This could perhaps shed some more light on what they encompass and where they lack.

3. Conclusion

This is still very much a work in progress where minimizing error is a major concern. As this is simply a service layer it is important to create a valuable interface with meaningful metrics so that traffic shapers, routing protocols and admission controllers can achieve a significant boost. Otherwise the effort is only of theoretical interest. In addition we are considering augmenting the interface with received signal strength indicator and expected transmission count to give a better picture of link quality.

4. Acknowledgements

I would like to thank Lars Landmark for providing comments on my work and the demo proposal, and Frank Li for his insights to the active probing technique. Also I would like to especially thank Mathieu Deziel for being so forthcoming with regards to the passive estimation technique.

References

[1] G. Anzinger. High resolution POSIX timers. http://sourceforge.net/projects/high-res-timers.

[2] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE journal on selected areas in communications*, 18(3), 2000.

[3] L. Chen and W. B. Heinzelman. Qos-aware routing based on bandwidth estimation for mobile ad hoc networks. *IEEE journal on selected areas in communications*, 23(3), March 2005.

[4] M. Deziel and L. Lamont. Implementation of an IEEE 802.11 link available bandwidth algorithm to allow cross-layering.

[5] S. Leffler, M. Renzmann, and G. Chesson. Multiband atheros driver for wifi. http://sourceforge.net/projects/madwifi/.

[6] F. Y. Li, O. Kure, P. Spilling, M. Hauge, and A. Hafslund. Estimating residual bandwidth in 802.11-based ad hoc networks: An empirical approach, September 2004.