Journal of Network and Computer Applications ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Context- and social-aware middleware for opportunistic networks

C. Boldrini, M. Conti, F. Delmastro*, A. Passarella

IIT Institute-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

ARTICLE INFO

Article history:
Received 18 November 2009
Received in revised form
11 February 2010
Accepted 18 March 2010

Keywords:
Middleware
Context
Social-awareness
Opportunistic networks
Content sharing

ABSTRACT

Opportunistic networks are multi-hop ad hoc networks in which nodes opportunistically exploit any pair-wise contact to share and forward content, without requiring any pre-existing Internet infrastructure. Opportunistic networks tolerate partitions, long disconnections, and topology instability in general. In this challenging environment, leveraging users' mobility represents the most effective way to deliver content to interested users. In this paper we propose a context- and social-aware middleware that autonomically learns context and social information on the users of the network, and that uses this information in order to predict users' future movements. In order to evaluate the proposed middleware on a realistic scenario, we have designed and implemented a context- and social-aware content sharing service, exploiting the functionality of the middleware. Both the middleware and the content sharing service have been integrated with an existing data-centric architecture (the Haggle architecture) for opportunistic networks. Finally, we have validated the proposed content sharing application on a small-scale testbed and, on a larger scale, we have investigated the advantages provided by context- and social-aware sharing strategies by means of extensive simulations. The main result of this paper is the definition and implementation of a context- and social-aware middleware able to share context information with all the interested components improving the efficiency and performances of services and protocols in opportunistic networks. With respect to content sharing strategies that do not exploit context and social information, we have obtained up to 200% improvements in terms of hit rate (probability that users receive the content they request) and 99% reduction in resource consumption in terms of traffic generated on the network.

1. Introduction

Opportunistic networks represent one of the emerging communication paradigms for pervasive and ubiquitous environment by supporting wireless communications in intermittently connected scenarios. We can foresee in the near future that the number of mobile devices with networking capabilities carried by users or available in the environment will be huge. Assuming that all of them will be covered at any time by high-bandwidth wireless infrastructure services is questionable. It is likely that the network will be composed of clouds of wireless devices that appear, disappear and reconfigure dynamically, while events such as disconnections of those clouds from the global Internet, their dynamic reconfiguration and partitions of the network will be the rule rather than the exception. Some of these clouds can include devices connected to the legacy Internet (e.g., through WiFi Access Points), while some others can be (temporarily) disconnected from the rest of the network. Most of the time, such clouds exist because groups of users spend time in the vicinity of each other. Therefore, the existence and evolution of clouds is related to the existence of social communities of users, that spend time together because they have either temporary or long-lasting social relationships (e.g., people shopping in the same mall, coworkers, or the members of a family).

Opportunistic networks propose an innovative communication paradigm to cope with this kind of scenarios, by exploiting users' mobility to create opportunities for communication among nodes when global connectivity is not available, or communication through a globally connected network is not the most efficient solution (e.g., because the communication endpoints are in the vicinity of each other). From this standpoint, opportunistic networks complement conventional solutions based on wireless infrastructures, such as cellular and WiMAX. Opportunistic networks represent a natural evolution of Mobile Ad hoc NETworks (MANETs) (Conti and Giordano, 2007), maintaining the basic features of cost-efficiency and self-organization, as devices still self-organize in order to build multi-hop ad hoc networks without requiring any pre-existing infrastructure. However, they completely re-design the characteristics of networking protocols proposed in MANETs, making them able to support the absence of a stable path between pairs of nodes that wish to communicate. Indeed, while mobility in MANETs is seen as a negative feature

* Corresponding author. Tel.: +39 050 3152405; fax: +39 050 3152593.
E-mail address: franca.delmastro@iit.cnr.it (F. Delmastro).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.03.017

which makes the network topology unstable, in opportunistic networks it is exploited as it generates additional opportunities for nodes to get in touch and communicate. One of the cases that best exemplify the bottom-line concept of the opportunistic networking paradigm is multi-hop forwarding (see, e.g., Hui et al., 2008; Spyropoulos et al., 2008; Daly and Haahr, 2007; Zhao et al., 2004; Wang et al., 2007; Balasubramanian et al., 2007). In this case multi-hop paths are built dynamically, on-the-fly, by intermediate nodes (i.e., mobile relays) that evaluate all the current opportunities to forward a message to their neighbors in order to reach the final destination. Thus, they must be able to store the messages when no forwarding opportunity exists (e.g., no other useful nodes in the transmission range), and send them as soon as a new opportunity arises. For this reason the forwarding paradigm is referred to as ‘‘*store-carry-and-forward*’’ (Fall, 2003).

The opportunistic networking paradigm is suitable to address several problems other than multi-hop forwarding. The concept of opportunistic networking is gaining momentum in mobile content dissemination systems, due to the increasing use of mobile devices as tools to generate and share content with (nearby) users. For example, as shown in Fig. 1 users' devices can be sources of information relevant to other users (e.g., a picture generated by a user camera or a local map available on a PDA) and this information can be distributed to other users via Internet (if available) and/or through the store-carry-and-forward paradigm by exploiting the nodes' mobility.

Recently, the opportunistic networking concept has also been identified as a key building block of two very interesting paradigms, namely opportunistic computing (Conti and Kumar, 2010) and people-centric sensing (Campbell et al., 2008). According to the opportunistic computing concept, users shall exploit resources available on devices spread in the environment (including mobile devices of other users nearby) to create rich composite services that could not be built exploiting local resources only. Opportunistic networking clearly provides the natural communication paradigm of such an environment. The concept of people sensing centric starts from the pervasive availability of smart devices featuring multiple sensors (e.g., those available on smartphones such as cameras, microphones, accelerometers). Based on sensory readings collected from a multitude of devices, it is possible to infer the status of the users and their current activities, and exploit this information to create

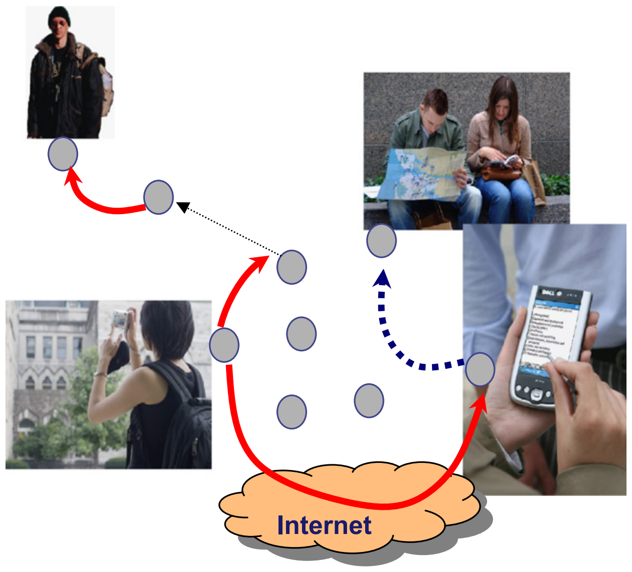


Fig. 1. User-generated content and opportunistic networking.

enhanced mobile social networking services (Miluzzo et al., 2008). Opportunistic networking techniques can be used as the communication substrate to collect and share sensory readings in this case. Other applications areas that will benefit from the opportunistic networking concepts include pervasive healthcare, intelligent transportation systems, crisis management (see Conti and Kumar (2010), Roccetti et al. (2007), and Lee et al. (2009) for more details and examples).

Irrespective of the particular application area, an important feature of opportunistic networking solutions is exploiting context information describing the environment and the users, and in particular information about the users' social relationships. In general, topological information is in fact quite unstable and not very reliable in this types of networks, and therefore it should be complemented with context and social information to obtain a better understanding of the evolution of the contacts among users, and therefore of the near-future opportunities of communication (Silvis et al., 2006). It is thus clear that collecting and managing context information is of paramount importance in this environment. In particular, a general approach is required, which could provide context management as a common feature that the different opportunistic networking services (routing, content dissemination, etc.) can exploit to optimize their behavior. Designing and evaluating such a context management solution for opportunistic network is the main contribution of this paper, as described in the following section.

1.1. Contribution

The key contribution of this paper lies in investigating some of the key architectural challenges of building a *social*- and *context-aware* middleware for opportunistic networks. Context management is not a new topic in the mobile and distributed networking area; however, the new challenges set by opportunistic networks call for novel approaches and architectural solutions. Conventional middleware designs for context management assume that the mobile network underneath is (to some extent) stable, thanks to lower-layer protocols (e.g., routing, mobility management, etc.) that mask mobility, disconnections and instabilities of all kinds. Middleware architectures for opportunistic networks cannot build on such assumptions. Information traditionally pertaining to lower levels of the stack (such as link availability, contact opportunities, communication costs) becomes fundamental also for the middleware operations, and lower layers can also benefit from context information managed by the middleware. The approach we propose in this paper is that of a clean-slate architectural design, in which the rigid separation among layers is replaced by a more fluid (yet modular) infrastructure allowing for more effective optimizations. As a specific example, we integrate our context-aware middleware inside the reference architecture of the European Haggle project,[1] which studied one of the most advanced architectural solutions of this kind. In the paper we consider, as a realistic scenario of the application of our social-aware middleware, a content sharing service that exploits user social relationships to optimize the distribution of data so as to maximize the probability of delivering content to interested users (Section 4). Both the middleware and the content sharing service have been implemented in a real prototype to validate its integration in the Haggle reference architecture (described in Section 5). The performance of the middleware and the content sharing service has been evaluated both through experiments (Section 6) and by means of simulations (Section 7).

[1] http://www.haggleproject.org.

The proposed context-aware middleware platform supports multiple services, ranging from low level protocols to user-level applications, as those described in our previous works (Boldrini et al., 2008a, 2008b). As a field assessment we use our middleware to provide context-awareness to a content sharing application for opportunistic networks presented in (Boldrini et al., 2008a). In this paper we just survey the different strategies proposed in (Boldrini et al., 2008a) and we use them to highlight the importance of context-awareness in opportunistic networks and the need of a general platform for the collection and management of context information. Differently from our previous works, in this paper we present a real implementation of our context-aware middleware and of the content sharing service within one of the reference networking architectures for opportunistic networks. We also validate our implementation through a small-scale real testbed, which also is used to provide preliminary experimental results for the first time with respect to our previous works on opportunistic networks. Finally, extensive simulations results are presented to complement and extend experimental results.

2. Related work

In this paper we present a social- and context-aware middleware for the collection, management and provisioning of context information in opportunistic networks. This substantially differs from the standard definition of middleware in distributed systems (Tanenbaum and Van Steen, 2002). In classical Internet architectures, middleware solutions are represented as an intermediate layer between transport protocols and applications, and they are generally aimed at the creation of overlay services that exploit standard communication protocols residing at lower layers to optimize upper layer applications, e.g. providing mechanisms for efficient distribution and recovery of data in highly distributed environments (e.g., mobile p2p paradigm). Wireless networks for pervasive environments maintain the need to define similar services and several solutions have been defined in the last few years for MANETs (see Chapter 23 in Bellavista and Corradi (2007) for details). However, these are not suitable for opportunistic network scenarios mainly due to the main characteristics of intermittent connectivity and lack of stable paths. In addition in this case, to improve the overall performances, all the network components need to be optimized and they can obtain enormous advantages from the interaction with a middleware platform dedicated to the management of context information. This typically calls for innovative solutions also from an architectural standpoint, as functionalities, traditionally seen as residing at the middleware layer, should be made available throughout the whole traditional stack. This results in cross-layer, or even layer-less architectures allowing all networking functions to exploit social context information (see some examples in Section 5).

Despite the importance of context information, all the studies presented so far are based on general descriptions of context information, without specific references to information models or software architectures for context management. Actually, in literature several studies on context-awareness in information systems have been conducted and several models have been studied (Strang and Linnhoff-Popien, 2004; Henricksen and Indulska, 2006; Schmidt, 2006). Initially, the definition of context mainly dealt with the interaction between the user and the applications, and all the information related to the status of entities (i.e., persons, places, objects) involved in that interaction (Dey, 2001). In the last few years, with the widespread use of pervasive devices and the emerging of new networking paradigms able to establish communications anywhere and anytime, the context definition has evolved, including several other factors (e.g., surrounding environment, interactions between users, interactions between devices of different users) considerably increasing its dynamicity and the number of involved entities. One of the models that better fit the continuously changing environment of opportunistic networks has been presented in Henricksen et al. (2002) and subsequently implemented in Henricksen and Indulska (2006) and is based on the definition of some general features of context information and their representation in a software system. Specifically, temporal constraints, level of accuracy/imperfection of the information, the levels of abstraction with which the same information can be represented, and the source of information are essential data to categorize and define a model of such wide range of data. The model proposed in Henricksen et al. (2002) provides a formal basis for representing and managing all these properties following an object-oriented approach that can result in a entity-relationship (ER) model for distributed environments, able to integrate the formal description with the qualitative evaluation of the context information (see Henricksen et al. (2002) and Henricksen and Indulska (2006) for details). This represents the strength of this model, compared with several others presented in literature, based on ontology or markup schemes (see Strang and Linnhoff-Popien (2004) as a survey on this topic). Thus, we can consider it as the reference solution for context modeling in the design of our social- and context-aware middleware for opportunistic networks. Before showing in the next section how the specific categories of context information can be modeled following these principles, in the following we briefly overview the state-of-the-art on networking protocols for opportunistic networks, in order to highlight the importance of context- and social-awareness in the design and optimization of such protocols. Routing protocols represent one of the most studied issues in opportunistic networks. Several solutions have been proposed in literature and they can be categorized based on the amount of information they leverage in order to autonomically learn the features of the network they are immersed in. These protocols can be classified as *context-oblivious* protocols, *partially context-aware* protocols, and *fully context-aware* protocols. On the one end of the spectrum, in pure dissemination schemes, nodes are oblivious to any available context information. In order to reach the message destination, they just rely on aggressively spreading the messages in the network, at the cost of a huge amount of data transmitted and stored at any node. This is the case of Epidemic Routing (Vahdat and Becker, 2000), in which copies of the messages to be delivered are sent to each encountered node during a pair-wise contact. On the opposite end of the spectrum, fully context-aware schemes (e.g., HiBOp (Boldrini et al., 2008b), BubbleRap (Hui et al., 2008)) leverage user-level social and context information, such as friends, coworkers, hobbies, etc., to selectively identify good next hops towards the destination of messages. Specifically, in HiBOp, at each node the context is built by collecting information on the encountered nodes, the visited places, etc. Then, based on the knowledge acquired, each node is able to compute its likelihood of meeting the message destination. To this aim, the most suitable next-hop is the one whose context is more similar to the context of the destination of the message. The message is then passed from node to node, following a path with increasing delivery probability, until the final destination is reached, or the message has expired. Between context-oblivious and fully context-aware routing approaches, we have partially context-aware protocols (like PROPHET (Lindgren et al., 2003) and Spray&Wait (Spyropoulos et al., 2007)) which rely on an simplified version of the context, usually made up of just one or two pieces of information. Typically, these protocols measure the frequency of direct contacts between nodes, and for this reason they are also called mobility-based protocols. In PROPHET, for example, the information on the nodes' past encounters is used to estimate the delivery probability, under the assumption that the trend of future encounters shares strict similarity with what has happened in the past, ignoring social information about the users. In Boldrini et al. (2008b) the performances of three protocols representative of the aforementioned routing approaches (context-oblivious, partially context-aware, and fully context-aware) have been compared by means of simulations. Here fully context-aware protocols proved to be able to drastically reduce resource consumption in terms of the traffic injected in the network and the buffer occupation. In certain configurations, this is paid with an increment in the delivery delay and message loss, which is however tolerable for typical delay tolerant application.

Recently, content dissemination systems have also been studied for opportunistic networks. Typically, they are variations of the publish/subscribe paradigm: publisher nodes generate contents and inject them into the network, subscriber nodes declare their interest in receiving certain types of content (e.g., sport news, radio podcast, blog entries, etc.) and strive to get it in some ways. Usually, nodes can be publishers and subscribers at the same time. Content distribution systems can be classified following an approach similar to that used for routing protocols. Social-oblivious content dissemination protocols do not use information on the sociality of nodes when deciding how to spread content items in the network. In this category, again the simplest approach is just to flood the network with multiple copies of the same items, which are distributed to any node upon contact, without taking into account the actual interests of the users (Vahdat and Becker, 2000). The TACO-DTN protocol (Sollazzo et al., 2007) relies on the presence of special nodes (infostations) in the network, that take care of disseminating the content items based on the interests of the passing-by users. In the PodNet architecture described in (Lenders et al., 2007) simple heuristics (e.g., random dissemination, disseminate most popular first, etc.) are proposed for selecting the subset of contents that two nodes should exchange upon contact. PodNet policies however do not take into account the social characteristics of the users of the network, like, for example, the fact that users tend to group into communities, or the fact that the users' travelling patterns across different communities can be very different from user to user. This might lead to suboptimal solutions with respect to the actual distribution of interests across communities and calls for advanced solutions that exploit the users' social dimension. One of the first social-aware publish/subscribe system proposed in the literature is the one in Yoneki et al. (2007), where an overlay structure is built using information on the social relationships among the users in the network. Specifically, the most central (from the social network standpoint) nodes are used as broker nodes in the overlay. Another social-aware content dissemination protocol (ContentPlace) is presented in Boldrini et al. (2008a). Specifically, its policies will be described in detail in Section 4, since they will be used as a field test for the context- and social-aware middleware proposed in this paper. Here we anticipate that ContentPlace drives the content dissemination process using the knowledge on the distribution of interests among the visited communities. A similar approach is also used in SocialCast (Costa et al., 2008).

In opportunistic networks, the problem of content dissemination shares many similarities with multicasting. Multicasting involves one-to-many communications in which the source and destination nodes are known to each other. Recently, the exploitation of social information has proved very effective also for the multicast problem. As an example, in Gao et al. (2009), multicasting is implemented as a multi-copy scheme, where the selection of the minimum number of relay nodes that guarantees a predefined delivery ratio is performed using the concept of node centrality (in the social network) and social community structure.

What emerges from the above overview on the state-of-the-art on opportunistic networking is the central role of context and social information, used at different levels of the networking stack, from routing to application. The reason is that, when no other more traditional network properties (topology, connectivity, etc.) can be exploited to find the paths from the source to the destination of a message, predicting the future behavior of the users of the network by means of context and social information can be the only way to deliver messages in an opportunistic network. Thus the presence of a middleware that collects and provides information on the context and the sociality of users is of paramount importance for supporting communications in this environment.

3. Context definition for opportunistic networks

Even though the definition of context should be so general to include all possible information in a general scenario, the design of a context-aware middleware platform for opportunistic networks requires a detailed definition of the specific involved information in order to provide an efficient management on constrained devices, and an effective access to this information by all the system components that can benefit from their use.

As previously explained, in pervasive and highly mobile environments the classical definition of network of (mobile) devices must be extended to the novel concept of several networks of people, where connections mainly reflect the social interactions among users that move around cities, places, and events carrying their mobile devices, and establishing thus opportunistic communications. In this view, we define the context as composed of three parts: (i) the *user* context, (ii) the *service* context, and (iii) the *device* context.

3.1. User context

The *user* context refers to two separate sets of information. The first one is mainly related to personal data of the user, like name, home address, work address, habits, timetables, most visited places, interests in specific applications or services. The second one mainly deals with the *social* characteristics of the user in terms of social contacts with other people, characterized by specific interests and preferences and the affiliation to specific social communities. The elaboration of these two sets of information and their relationships provide to the system a snapshot of the current connections among users and a probabilistic analysis of their network of contacts in the next future. In fact, according to social network theory (Watts, 1999), we can assume that users are grouped in communities, and users of the same community have strong social links between each other, usually related, e.g., to common interests, habits, and visited places. Each user is generally represented as belonging to a "*home*" community in which she spends more time, and is characterized by a *centrality* parameter as the popularity degree within the community in terms of number of interactions with the other members. Some nodes can also have social links outside their "home" community, modeling social relationships with users of different groups.[2] Thus, these users (also defined as *travelers*) generally act as bridges between disconnected communities and can be exploited to move data and information from one community to the others they are in contact with, exploiting additional parameters like *service* and *device* contexts (defined in

[2] Small-world theories have shown that these "external" links act as shortcuts between communities, thus enabling communications across the network through a small average number of hops (6 hops in the "classical" small-world model derived from the work by Milgram (1967), less than $O(\log n)$, where n is the size of the network, in several types of real networks (Newman, 2003)).

the following). Another important aspect is how to identify and detect social communities starting from the analysis of simple user's movements and contacts. In the last few years several algorithms for communities detection have been presented in literature (Yoneki et al., 2007; Hui et al., 2008; Danon et al., 2005; Newman, 2004), and they are mainly based on the analysis of contact duration and number of contacts between pairs of users, assuming that individuals meet at a higher rate if they have one or more mutual friends. Thus, adding to this information the personal and social information of each user, the middleware extends the notion of community with user's behavioral model, thus including the relationship between the presence of a user inside a community in a certain period of time, and the main reasons for that (e.g., she is in the gym because it is Friday afternoon and this is a user's habit), defining additional relationships in the social graph, both in terms of temporal information (i.e., how long people are in contact) and the social role of the user in that community (e.g., friends, family components, colleagues). The entire graph provides thus a complete analysis of the user context, allowing probabilistic analysis to predict future contacts of each single user.

3.2. Service context

The *service* context strictly depends on the service or application currently running on the mobile device and with which the user directly interacts. In fact, each service category (e.g., content sharing, chat, data dissemination) is characterized by specific information that can improve its performance. For example, in case of content sharing service, the interests of users in sharing specific file categories, or contents identified by user-defined attributes, can be used, in addition to previous social context information, to identify possibly interested users and to select, in an opportunistic way, the appropriate next-hop to forward a specific data that have to be delivered to a specific destination. These pieces of information, even if some of them are directly requested to the user, e.g., through a graphical interface for the service configuration, are not related to the personal information of the user and they are generally represented by the same kind of information used by traditional p2p systems. Thus, they actually might not be considered as sensitive data from a privacy standpoint. However, encryption mechanisms can be included in the context management system to restrict the access to this data only to trusted users (e.g., members of the same community). Currently advanced solutions to cope with security and privacy issues in this scenario are under investigation (Boldrini et al., 2008b; Shikfa et al., 2009).

3.3. Device context

The *device* context is mainly related to physical characteristics and limitations of mobile devices, in terms of capacity, battery life, and embedded technologies (e.g., wireless interfaces, cameras, sensors). In fact, current devices are characterized by several attributes that can be considered as additional context information in the management of an opportunistic network. First of all, information related to the wireless interfaces, currently active on each device, can further improve the network connectivity, making the system able to adapt the network configuration of the single devices to the surrounding wireless capabilities. In addition, information on the current status of the storage capacity or battery life of the device can have a strong impact on the communications and data exchange between nodes. Thus, the device context mainly deals with resource management issues in opportunistic networks that, if correctly analyzed and related with the other contexts, can enhance both network's and applications' performances.

Note that all the context information identified by these categories can be represented through the formal model described in Section 2. Specifically, each single entity (e.g., users' interests, capacity, battery life, user's personal information) is characterized by a temporal validity, a level of accuracy, and a level of abstraction chosen to represent it. For example, considering a content sharing service, the users' interests can be considered valid for a certain period of time; they are completely accurate, since they are generally selected by the user herself; we can use the list of genres of files the user prefers to share with her friends as the level of abstraction to represent their values. As far as the possible sources of context information in opportunistic networks, we must consider that the definition of the previous three different notions of context directly involves different components of the node architecture (e.g., user's interaction, service features and device characteristics), but also the interaction with other nodes of the network in order to have a view of the context surrounding the local user. To better understand how the collection and management of context information impacts on the overall system architecture, we present in the next section an application scenario that tries to satisfy the need of people to generate and share contents anytime and anywhere. It applies main principles of the User-Generated Content model, defined by the Web 2.0 paradigm, in pervasive and highly mobile environments and makes them available based on context information.

4. A realistic scenario: social- and context-aware content sharing service

In this paper we propose a social and context-aware content sharing service to highlight how using context and, in particular, social information derived from mobile users and devices allows us to efficiently provide this service even in intermittent connectivity scenarios. In this section we describe the design of the service, how it exploits context information describing social relationships among users, and how it can interact with a middleware platform for context management integrated in a reference architecture for opportunistic networks like that proposed in the Haggle project.

The main idea of the content sharing service is to exploit traveler nodes to establish opportunistic communications between separate communities, making all the users able to share and disseminate contents even with those users that they will never get in touch with. Traveler nodes, moving from a cloud to another, define sporadic "off-line" connections among clouds. Typically, travelers are users with social relationships with more communities, and thus they can be exploited as bridges among clouds, since it is very likely that, upon leaving a community, they will visit it again after a while. Thus, the traveler node is used as a sort of *data mule* that moves from a community to another, selects interesting content for nodes belonging to different communities and, if needed, makes its own resources available to download and subsequently transfer content to the interested users. To this aim, the service must exploit context information related to both the local node and all the encountered users belonging to separate communities. In this way it will be able to define a ranking of the available content items with respect to a measure of the utility of that content for the local user and for the others participating in the same service. The core of the service operation is indeed the definition of functions to rank content items based on several parameters (e.g., users' interests, physical constraints, probability of encountering a selected user in the next future). Details on the definition of this metric will be described in the next section.

Before analyzing formal models and technical details, it is worth pointing out which type of context information is required by the content sharing service, with respect to the general classification provided in Section 3. We start by discussing which information we need to build the *service* context. Content items are generally represented by files stored on users' mobile devices, and they can be characterized by a set of attributes defined autonomously by the service (e.g., genre, type, size) or by the user to specify additional information related to the content of the file (e.g., a picture of Paris). The same information can also be used by users to declare their sharing interests, together with the list of files they decide to share with the others through the service. This information basically constitutes our service context. Further information like user's habits, personal data, mobility patterns, and all data related to social behaviors and contacts of the user are managed independently of the service as *user* context. As far as the *device* context, we decided to consider only the local storage capacity as physical constraint of the system, in order to simplify the design and development of the service. The *user* and *service* contexts must be disseminated in the network to make the other users aware of interests and contents shared by their neighbors, thus allowing the service to optimize its features exploiting this information. This is done including this information in beacon messages exchanged by nodes during the neighbor discovery procedure. Since the most part of this information requires a direct interaction with the user, a user-friendly graphical interface is developed on the mobile device as shown in Fig. 2. After the user has inserted the needed information, all data is passed to the middleware platform in charge of storing and elaborating it before the transmission on the network. Technical details on the structure of the context information and the interactions between the middleware and the other network components will be described in Section 5.1, analyzing the integration of the service in Haggle architecture.

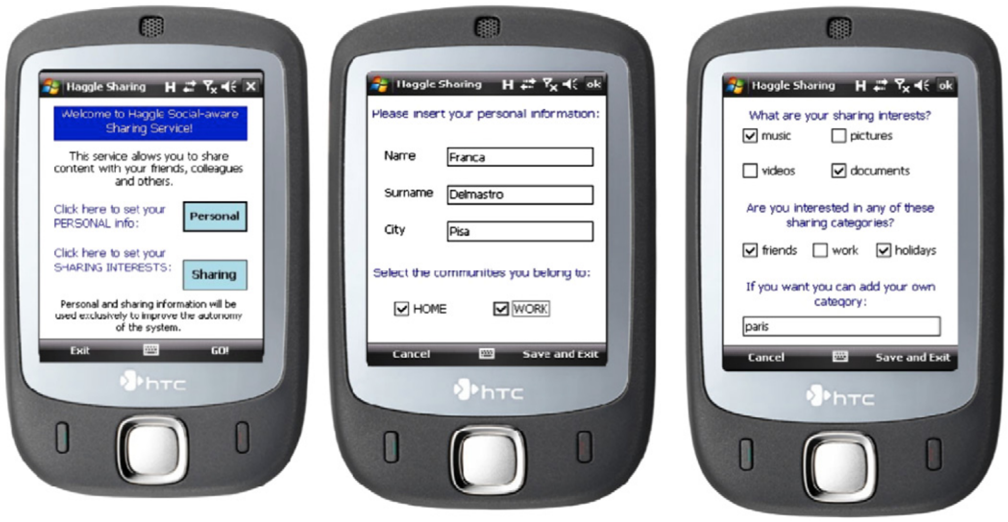
The user and service context information is the basis to compute the ranking among content items. As will be clear from the description in the following section, computing ranking requires that each node maintains a continuously updated snapshot of the surrounding users' context, as well as a historic profile of the encountered context information. To this aim, two separate data structures are defined on each node: the *current context* (i.e., context information associated with the nodes of the community the user is currently in touch with), and the *context evolution* over time (i.e., information about context of past encountered users as the evolution of social contacts of the local user). In the next section we describe a content sharing model that defines the utility of a specific content based on the social behavior of the involved users.

4.1. A social-oriented utility-based framework

The key element of the optimization of the content sharing service is a *utility*-based framework that drives the dissemination of content in the network. Specifically, the framework allows each node to rank content items encountered on other peers, and decide which items should be fetched from encountered peers because it is (likely) to be interesting to other users that will be encountered in the near-future with high probability (because they belong to the same social community). According to the framework, nodes assign a utility value to each content item, and then pick items as to maximize the total utility of the data they will store locally. The core of the framework is thus the definition of the function that nodes use to assign utility values to content items. For each node, the utility not only considers the interests of the local user, but also considers the expected utility of a content item for the other members of the user social communities. Under the assumptions that users can belong to one or more social communities, we can thus define the utility function of content c as follows:

$$U(c) = u_l(c) + \sum_{i \neq l} \omega_i u_i(c) \qquad (1)$$

where $u_l(c)$ is the utility of a specific content for the local user, $u_i(c)$ is the utility for the ith community the user is in contact with, and ω_i is a cooperation index (weight) that defines the willingness of the user to cooperate with the ith community (i.e., to spend its own resources to increase content availability for that community). In Eq. (1) we have stretched a bit the concept of community by representing the local user just as another community the user is in touch with, as such it is associated with a utility value of its own. We assume that a user is always willing to cooperate with itself, i.e., to retrieve the content it is directly interested in. All the other communities can be mapped into whatsoever aggregation of users, ranging from simple co-location (e.g., users that are currently in the radio range of each other) to pure social communities (e.g., community of the family members or community of friends). Note that the definition of the weights determines the social behavior of the local user. For the


Haggle Sharing

Welcome to Haggle Social-aware Sharing Service!

This service allows you to share content with your friends, colleagues and others.

Click here to set your PERSONAL info: Personal

Click here to set your SHARING INTERESTS: Sharing

Personal and sharing information will be used exclusively to improve the autonomy of the system.

Exit GO!

Haggle Sharing

Please insert your personal information:

Name: Franca

Surname: Delmastro

City: Pisa

Select the communities you belong to:

☑ HOME ☑ WORK

Cancel Save and Exit

Haggle Sharing

What are your sharing interests?

☑ music ☐ pictures

☐ videos ☑ documents

Are you interested in any of these sharing categories?

☑ friends ☐ work ☑ holidays

If you want you can add your own category:

paris

Cancel Save and Exit

Fig. 2. Context-aware content sharing service.

reader's convenience, we discuss the different policies that can be used by customizing the definition of the weights in a dedicated Section 4.2.

The utility function is made up of one component for the local node, and one component for each community available in the environment. The definition of the utility components is unique. Specifically, the utility of a content item for a generic community i is defined as the access probability of the users of community i to the content item (p_{ac}) multiplied by a function f_c defining the cost of accessing the item for those users, divided by the object's size (s) (Eq. (2)). This definition of the utility function is consistent with the forms used in other domains where utility-based frameworks are applied, such as Web caching (Balamash and Krunz, 2004):

$$u_i = \frac{p^i_{ac} f_c(p^i_{av})}{s} \tag{2}$$

Generally, we consider the *access probability* as the probability that either the local node (in $u_l(c)$) or the nodes of community i (in $u_i(c)$) will be interested in downloading the specific content. The cost function f_c is considered to be dependent on a single parameter only, i.e., the availability of the content item in community i. The function f_c is a monotonically decreasing function of the availability p_{av} (measured as the percentage of community members already having a copy of the content item). The rationale of this choice is that the marginal utility of fetching a content item should monotonically decrease as that item becomes more and more spread in the community. Specifically, we use an exponential function as cost function since it achieves a fairer behavior with respect to a linear decaying function in a preliminary study that we have performed using the simulation environment presented in Section 7 (see Boldrini et al. (2008a) for more details).

4.2. Social-oriented policies

As already anticipated, the ω_i parameters express the willingness to cooperate with the other nodes of the communities present in the network. By properly defining ω_i, we can implement a number of different policies that exploit users' social relationships. In this paper we study five such policies. In the first three, users cooperate with both the community they are currently in touch with and the communities that they are likely to meet in the future. In the last two, we decided to explore the case of traveler users completely projected to the future. In the latter case, the idea is that cooperation helps when diversity is high. Therefore, users should proactively download files for nodes that they will meet in the future (and that have currently a very different context) instead of allocating their resources also to help the users, with whom they are currently co-located, to spread content . In detail, the policies that we consider are defined as follows:

- *Uniform social (US)*: Under this policy, the weight ω_i is equal to 1 for all the communities visited by the traveler user. With this choice, there is no preferential selection of communities: all are equally served, when possible, by the traveler users, regardless of the frequency of their meetings.
- *Present (P):* With this policy, users download files that can only be interesting for the community where they are currently roaming or for themselves. Therefore, $\omega_{CC}=1$ and $\omega_i=0$, $\forall i \neq CC$, where CC represents the current community. With this strategy, every time a user changes community, it becomes an active member of the new community.
- *Most frequently visited (MFV):* The weight ω_i is assigned to community i proportionally to the time spent by the local user in this community, i.e., $\omega_i = (t_i/\sum_i t_i)$. This policy favors the communities with which the local user is more likely to get in touch.
- *Future (F):* The weights of all the communities apart from the current community are set as in MFV. Therefore, with this strategy users cooperate with the different communities in a way proportional to the time spent in each of them. However, here the weight of the current community is set to zero ($w_{CC}=0$) in order to force the traveler users to cooperate only with the communities that they will meet in the future.
- *Most likely next (MLN):* In this policy we condition the probability of getting in touch with a generic community i on the current "social" position of the local user. Translated in a more formal definition, in this policy $\omega_i = P(c_i | CC)$, where c_i is the ith community, CC corresponds to the current community, and thus ω_i represents the probability of visiting community i at the next step, given that the current community is CC. As in the future policy, the weight w_{CC} is set to zero, thus excluding cooperation with the current community.

These social-aware policies will be compared to two non-social policies: the Greedy policy, under which traveler users are willing to cooperate with other nodes only for the content in which they are directly interested, and the Uniform policy, under which nodes download content uniformly at random.

These social-aware policies require online, dynamic estimation of the social weights and the utility parameters, which basically are extracted from information on the users' contexts. Therefore, our context- and social-aware middleware can be fully exploited to enable a content sharing service that relies on these policies. As far as the parameters of the utility function are concerned, the middleware implements the collection and estimation of the access probability and availability of the content for each community visited by the traveler node. The access probability is estimated from the interests advertized by users during pairwise contacts and stored locally by the middleware. These values are then aggregated to obtain a statistic for the whole community. Upon subsequent contacts the access probability is updated according to the newly advertized values. The process is very similar for the content availability. When two nodes meet, they exchange a summary of the content that they currently have in their buffers. By storing this information, an aggregate measure of the availability of a given content for each community can be computed by the middleware platform. The interested reader is referred to Boldrini et al. (2008a) for more details on the estimation algorithms. The computation of the social weights requires a community detection mechanism that makes the middleware able to recognize the current context of the user. There are promising results about autonomic community detection systems (Hui and Crowcroft, 2007; Hui et al., 2007). However, being this topic orthogonal to our investigation, we simply assume that one of these mechanisms is in operation in the reference architecture for opportunistic networks in which the middleware will be integrated. Using the information provided by the community detection mechanism, our middleware is then able to estimate the time spent in each community and therefore to compute the social weights. More details on the techniques used for the estimation can be found in Boldrini et al. (2008a).

From the description of this social-aware content sharing system, it is clear the importance of a middleware that collects and manages context and social information and is able to implement a model that reflects the users' social behavior and the utility of the information to be shared and disseminated among users. To make this system a reality, we must integrate our middleware platform in a software architecture for opportunistic networks. For this reason, in the next section we analyze main features of innovative architectures defined for this new networking paradigm, with particular attention the solution proposed by the Haggle project.

5. Clean-slate architectures and context management

The novel requirements of opportunistic networks resulted in the proposal of innovative architectures that may even bring the cross-layer paradigm to the extreme of layer-less solutions. These architectures are usually called *clean-slate*, as they drastically depart from conventional Internet design paradigms. The reason is the high degree of instability and dynamism of mobile networks (which features links instability, long partitions, frequent disconnections and high churn rates) coupled with scarce resources availability (e.g. in terms of bandwidth and capacity) compared with legacy Internet. Due to the resource limitations, protocols at all layers (of the traditional Internet architectural model) can be significantly optimized by exploiting information that would remain hidden to them in conventional Internet designs. Therefore, researchers are investigating cross-layer solutions for opportunistic networks in which protocols can access a large set of information describing the networking environment and the current status of the network, without compromising the inherent modularity of Internet architectures. Examples of such architectural solutions can be found, for example, in the European Projects ANA,[3] BIONETS,[4] and Haggle,[5] as well as in US Projects such as Content Centric Networking at PARC (Van Jacobson et al., 2009)) and the Clean-Slate Program at Stanford.[6] In this paper we explicitly focus on the Haggle reference architecture as it explores a completely layer-less modular architecture for opportunistic networks. This is particularly suitable for our needs, as we can integrate our context-aware middleware within this architecture, and make its functionality available to all the other services, and specifically to the content sharing service that we use as a concrete test case.

The FET-SAC Haggle project, funded by the European Commission, aims at defining a new autonomic architecture for opportunistic networks and it envisions a *data-centric* architecture as the most promising approach. The main element of Haggle architecture is represented by the *DataObject*, a data structure aimed at containing any type of data, from messages to be sent on the network (both application and control messages) to internal data exchanged by single components. DataObjects are characterized by a set of attributes defined as a descriptive XML header composed of several metadata (i.e., a name-value pair for each attribute). The content item is then stored as a payload in the data structure. Thus, the DataObject always assumes the form of a message, both for internal and external communications; single Haggle's components must declare their interest in specific attributes to be notified of the arrival or generation of the related DataObject. The active components of Haggle architecture are defined as Managers and they are in charge of implementing specific functionality of the network architecture, from forwarding protocols, to resource management, connectivity management, security and so on. They mainly implement the basic features needed to communicate and deliver messages on the network, and they can make use of modules to implement specific functionalities related to particular algorithms (e.g., the routing protocol described in Boldrini et al. (2008b) can be seen as a module of the Forwarding manager). As shown in Fig. 3, Haggle architecture is originally composed of seven managers, each of them with a specific role:

- The Connectivity manager directly interacts with the physical layer of the architecture to detect all the available network interfaces and makes the system able to adapt to the current technology providing useful information on the network connectivity to the other managers.
- The Forwarding manager implements basic features of forwarding protocols for opportunistic networks, establishing routes on-the-fly, while data flows from a node to another in hop-by-hop communications. Specific functionality and optimization of forwarding protocols are then implemented as modules of the Forwarding manager.
- The Data manager is in charge of creating, storing and recovering DataObjects through the interaction with a central DataStore.
- The Resource manager implements basic mechanisms for local resource management, in terms of battery lifetime, capacity and other physical constraints of the local device to further optimize the node's behavior considering also its physical requirements.
- The Security manager implements security and privacy policies for the management and transmission of both applications and context data.
- The Attribute manager stores attributes related to DataObjects in the DataStore and maintains their relationships to identify interested managers and modules.
- The Protocol manager implements basic functionality of communication protocols for wireless opportunistic networks, mainly related to the transport layer. Specific features depending on the wireless technologies used (e.g., Bluetooth, WiFi) and the current state of the network are implemented by its modules.

All the information generated by each manager is organized and stored in the DataStore in the form of DataObject, specifying four main entities strictly connected among them:

- Interfaces: physical communication interfaces of the local device.

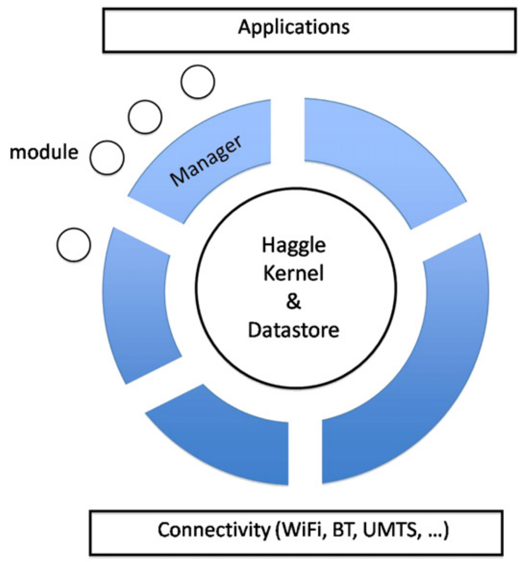


Fig. 3. Haggle node architecture.

[3] http://www.ana-project.org.
[4] http://www.bionets.eu/.
[5] http://www.haggleproject.org.
[6] http://cleanslate.stanford.edu/index.php.

- Attributes: set of name-value pairs describing data, users and devices.
- Data: virtual representation of persistent DOs such as application data; associated metadata is stored in the Attributes set.
- Node: virtual representation of a device or a user; this entity is directly connected with Interfaces entity to retrieve information about devices' physical interfaces, while it is connected with Attributes for descriptive information about devices and users.

Therefore, interactions among managers are related to the generation and update of DataObjects in the DataStore, implementing thus the concept of event-driven and data-centric architecture. To this aim, all the managers interact with a central entity: the *Haggle Kernel*, a minimal event queue that coordinates actions and communications between managers, and waits for incoming data. Managers subscribe to events declaring their interests in specific attributes and values. Events are then triggered by the kernel upon receiving corresponding DataObjects, both from the network and internal processing. The DataObject that triggered the event is handed over to the manager(s) that subscribed to it, which in turn process the DataObject according to its local policies. Note that this architecture is very extensible, as new functionalities can be added by simply defining additional modules for the managers. Exploiting this modularity, we have integrated the proposed context- and social-aware middleware as an additional manager, called Context manager, inside Haggle architecture.

5.1. Integration of the social- and context-aware middleware with Haggle

The Haggle node architecture natively supports context-awareness of all involved tasks due to possible interactions of managers through the generation and access to DataObjects. However, originally, it does not exist a manager dedicated to the management of context information. Therefore, we added a Context manager able to support all the other components (i.e., managers, modules and services) in improving their features by exploiting context information. This is quite natural in a data-centric architecture and especially in the Haggle architecture, where each message can carry with it additional information related to its content and its intended recipients in the form of metadata. However, the amount of context information that can be associated with a device, a user, and a service can overload the DataObject header, and may require also a priori knowledge from the other managers of specific context information to be able to declare their interest and be successfully notified. For this reason, the Context manager defines a "Context DataObject" in which it stores all the context information that must be exchanged among neighbors and that can be useful for the internal managers. This DataObject, collecting all the definitions of context described above, is spread among 1-hop neighbors through periodical beaconing, so that each node can maintain both local and surrounding context information. The Context manager will then implement probabilistic and qualitative analysis related to the context information based on explicit requests derived from the other managers. Based on the results of this analysis the other managers are able to take decisions in order to optimize their own performances.

Generally, the Context DataObject is maintained in the DataStore in the standard format but, since its definition involves different requirements of management depending on the interested managers, we must specify how the Context manager handles the three different context's components defined above. As far as the *service* context, a direct interaction with the service running on the device is required, since it can decide to maintain this information private to the specific instance of the service (i.e., only nodes running the same service can receive this information), or to allow even other services or Haggle's managers to exploit this information. Thus, the Context manager receives from the service the definition of its context and includes it in a DataObject dedicated to the spread of context among neighbors. If this context is private, only neighbors running the same service will receive that information, on condition that they have previously declared their interest in an attribute that generally defines the service identifier (e.g., content sharing, chat or others). As far as the *device* context, its information can be directly recovered by an interaction with the interested managers (e.g., Resource manager for capacity and battery lifetime, while Connectivity manager for active wireless interfaces), and it is generally interesting only for the local managers. Instead, the *user* context requires a direct interaction with the user to recover his/her personal data, and the authorization to use it in the system, while all the information related to its social contacts and communities are internally computed and evaluated through statistical analysis based on the exchange of context among neighbors. Therefore, the Context manager collects all the context information related to the local node and user and sends them as a Context DataObject through the periodical beaconing implemented by Haggle. When a neighbor receives that DataObject, the Context manager recognizes that it contains context information and appropriately notifies the other interested managers and services, before storing it in the DataStore. In this way each node is aware of the user and service contexts of all its neighbors.

This mechanism represents a general platform for context management integrated with the Haggle architecture on top of which new improved and optimized solutions for communications and content distribution in opportunistic networks can be developed.

In the next sections we validate the integration of the proposed solution in a real environment using HTC smartphones as mobile devices. The performance analysis derived from the real small-scale deployment give us important indications on the main requirements of the system in a realistic scenario (see Section 6). Then, to test the properties of the designed solution on a larger scale and with respect to a larger range of parameters, the performance evaluation has also been carried out by developing a simulation model. The main advantage of using simulation frameworks is the possibility of using a large amount of nodes and of defining accurately the involved parameters, guaranteeing the repeatability of the experiments. These conditions cannot be insurable in real testbeds, also due to the influence of several external parameters. Through the simulation model we are also able to investigate which of the proposed social-oriented policies implemented by the context-aware middleware are the most effective and efficient when applied to an opportunistic network, as it will be shown in Section 7.

6. Experimental validation and results

In order to validate the proposed context-aware middleware integrated with the Haggle architecture, we developed a prototype of both the Context manager and the content sharing service, integrated with a stable release of Haggle software. To easy the deployment of upper layer services, the Haggle architecture defines an Application Programming Interface (API), called libhaggleCS, that allows the interaction of applications and services with the main functionality of the Haggle managers (e.g., connectivity, forwarding, resource management and communication protocols). A demonstration of the main features of the system has been presented in the demo session of an international conference (Conti et al., 2009), directly involving a limited set of real users (i.e., up to five active smartphones with which users can interact by selecting their sharing interests). However, it is really expensive to set up a real testbed for opportunistic networks involving a large number of users, divided in a significant number of communities, to effectively validate all the social-oriented policies presented in the previous section. Furthermore, in a real testbed it is almost impossible to finely control the large set and wide range of parameters that can impact on the performance of our middleware. For these reasons, in the testbed we present a simplified scenario in which nodes in the same community are connected in ad hoc mode exploiting the wireless interface based on the IEEE 802.11 standard, and only the Most Likely Next (MLN) policy is developed by the Context manager. In this way, traveler nodes will exploit their own resources to favor those neighbors that will be most likely encountered in the next future. This policy will be shown to be one of the most effective and efficient in Section 7, where we analyze more complex scenarios with respect to that used for real experiments. In this case, we consider as the reference scenario two social communities (X and Y) as shown in Fig. 4.

In the example user C shares social relationships with both communities and therefore it visits them on a regular basis. Within the two communities users share some common interests: on the left side users A and C are interested in mp3 and jpeg files, while B is interested in jpeg and avi files; on the right side both users D and E are interested in jpeg and avi. Assuming that all nodes in each community are able to communicate with each other, after the neighbor discovery phase each of them knows the interests and the file list of the others, and thus accordingly updates its current context. Starting from the configuration in Fig. 4a, as C belongs to both communities, it is highly probable that she will move towards the right-hand-side community and can thus help distributing contents to the interested users of that community. When she reaches the right-hand-side community, and context information are exchanged through the 1-hop discovery process, she is able to select those files that are most interesting for previously encountered users, A and B, and then carry the files to the original community to satisfy their requests. To validate the software architecture we ran a set of real experiments in which we measured the average time needed to exchange a Context DataObject among 1-hop neighbors of the same community and the average file transfer delay inside the community varying the size of the content items users want to share. These measures constitute the building blocks to evaluate the performance of the systems in more complex scenarios, as they provide the time required to perform the basic operations of the Context manager and of the content sharing service. Specifically, they provide measures of the effective time needed by the Context manager running on the traveler node to exchange context information inside the current community and by the content sharing service to download interesting contents for past encountered users.

Considering that the average size of a Context DataObject, including the content-sharing context information (i.e., interests of the local user and list of shared files), is about 2 KB, we measured that the average time for two nodes to exchange it during the 1-hop discovery procedure of Haggle is about 0.5 s. Fig. 5 shows the time required by two 1-hop neighbors to exchange a file of variable size (from 28 KB up to 6.5 MB, as shown in Table 1). We measured the delay both at the level of the content sharing service (bars tagged as “Application”), as well as the corresponding delay measured from inside the Haggle architecture (bars tagged as “Haggle”). The former is the amount of time needed by the content sharing service to send a request to one of its neighbors and receive the selected file, while the latter is the delay measured by Haggle kernel as the essential time to successfully complete the sending on the network of the DataObject containing the entire file. Results are obtained as the average of 10 experiments, with confidence intervals computed with a 95% confidence level.

We notice that the average delay increases with the content size, reaching peaks of more than 73 s in case of 6.5 MB (a standard dimension for audio files). We can notice that for small content sizes the transmission delay measured from inside Haggle is negligible with respect to the transfer delay measured at the application layer, while it increases when the content size is greater than 500 KB. The difference between the transfer delays

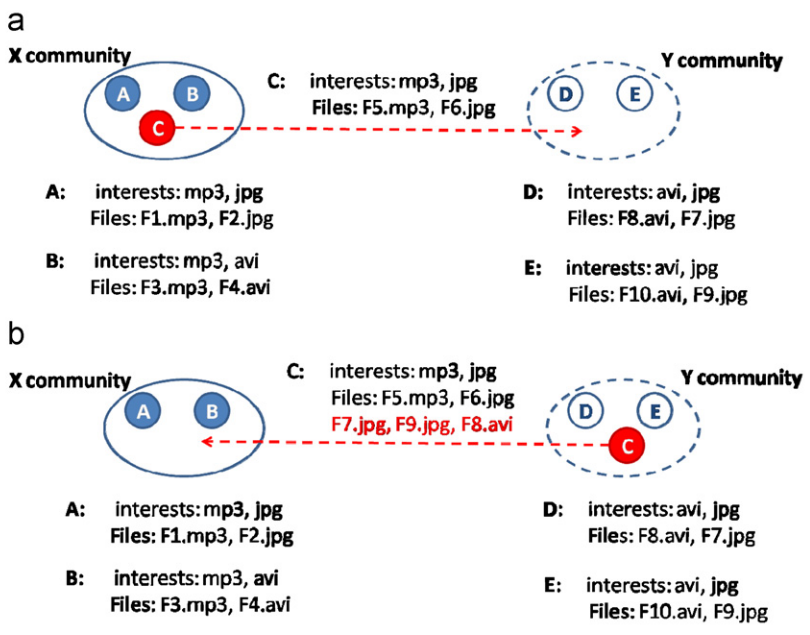


Fig. 4. Social-aware content sharing: application scenario.

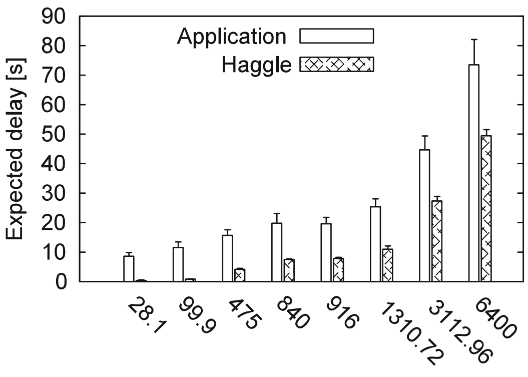


Fig. 5. Average file transfer delay.

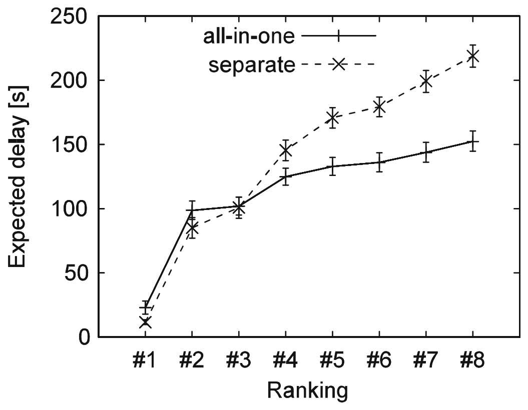


Fig. 6. Average transfer delay of single and cumulative requests.

Table 1
List of file IDs and size.

File ID	Size (kB)
#1	99.9
#2	6400
#3	475
#4	3112.96
#5	1310.72
#6	28.1
#7	840
#8	916

measured at the application layer and those measured by Haggle is mainly due to the internal management of incoming and outgoing DataObjects in Haggle kernel. In fact, since the kernel is implemented as a single event queue in which all generated DataObjects (both coming from the network or from internal managers' communications) are stored before being processed, the processing time needed by Haggle to satisfy a request depends on the status of the queue at the moment in which the request is stored in the queue.

To better investigate this behavior, we executed another set of experiments in which the traveler node sends a cumulative downloading request for all the available contents on the neighbor device. In Fig. 6 the average transfer delay of a cumulative request for the 8 files of Table 1 ordered in a random fashion is compared with the sum of the average transfer delays of each single download request measured in the previous experiment. In the cumulative request case, all files are requested simultaneously at the application layer ($t=0$ s). Fig. 6 shows the delay for each file received on the traveler node. Note that, for more than 3 requests, it is more convenient to send a cumulative request than single separate requests. This is due to the fact that a cumulative request is stored in the kernel queue as a set of subsequent single requests, while in case of separate requests, additional DataObjects, deriving either from the network or from internal manager communications, can be inserted in the queue, increasing the processing time of subsequent requests.

These experimental results highlight that, to develop a generic context- and social-aware service on top of the Haggle architecture enhanced with the Context manager, we must take into consideration both the time needed by the system to exchange context information and notify the upper layer service of the surrounding context, and the delay for contents' transfer that highly depends on their size. Therefore, to develop an efficient context-aware content sharing service in opportunistic networks, traveler nodes should evaluate the amount of time they spend inside a given community, as enough time should be allotted for accommodating the required content transfers. This is conceptually depicted in Fig. 7. Specifically, we can estimate that, to effectively download a set of interesting files from the Y community, node C should stay inside that community for a time greater than the sum of the average time needed for context exchange and the expected transfer delay of the selected files, using a cumulative request to reduce the processing time of the single requests.

The 1-hop transmission represents the basic scenario to exchange and share data, but its complexity further increases in case two nodes requiring multiple files to each other, generating thus a full-duplex traffic. Specifically, in this case each device has to manage both requests coming from the local application to be sent on the network, and those received from the network as content sharing requests to be served. This represents a different traffic pattern that can highlight the system ability to manage concurrent operations. Specifically, we ran several sets of experiments, progressively increasing the number of files requested by each node in a multiple request, from 2 up to 5 files requested in the same order used by the multiple request executed in the previous experiment. As shown in Fig. 8, the average transfer delay measured for each single file, mainly increases with the number of files included in the multiple request (i.e., the average transfer delay of each file lightly increases when it belongs to a multiple requests composed of an increasing number of files), but the measured delays result to be quite similar to those measured in the case of only one node sends a cumulative request towards the other node, as highlighted in Fig. 9. In this case, we considered a multiple request of five files and we compared the average delay to receive each file. As far as full-duplex requests, only the last two files measure an average delay greater than in the case of a cumulative request. However, there is no significant difference between the two cases, this demonstrates that the system is able to concurrently manage full-duplex requests without increasing the average transfer delays of single files.

In this last set of experiments we analyzed the reference scenario in which each node sends a cumulative request for multiple files, since we highlighted before that, for more than three requests, it is more convenient to send a cumulative request with respect to send single requests. However, a cumulative request is served by Haggle as a sequence of separate requests, and requests that are subsequently inserted in the queue suffer from the processing time needed to serve the previous ones. To better investigate this aspect we consider an intermediate scenario in which single requests are interleaved with a delay of

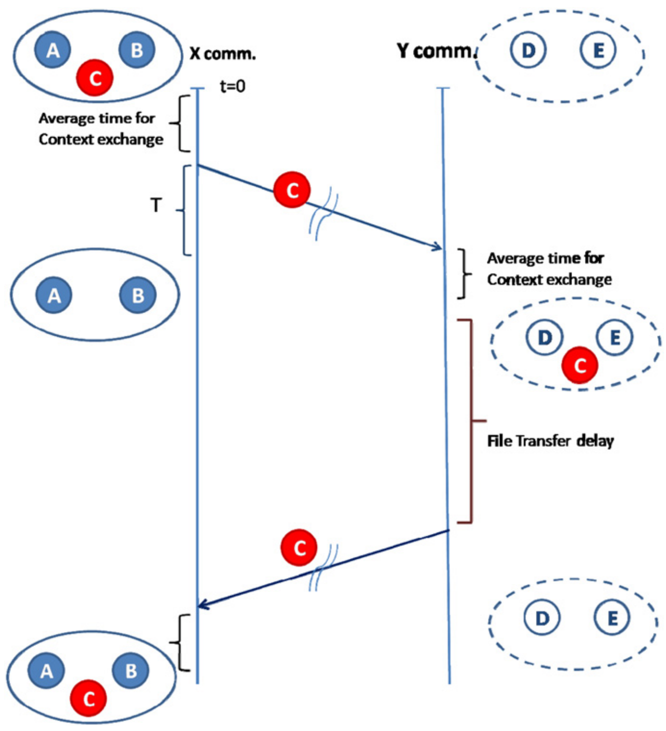

X comm.
Y comm.
t=0
Average time for Context exchange
T
Average time for Context exchange
File Transfer delay
A
B
C
D
E

Fig. 7. Sequence diagram of the real scenario.

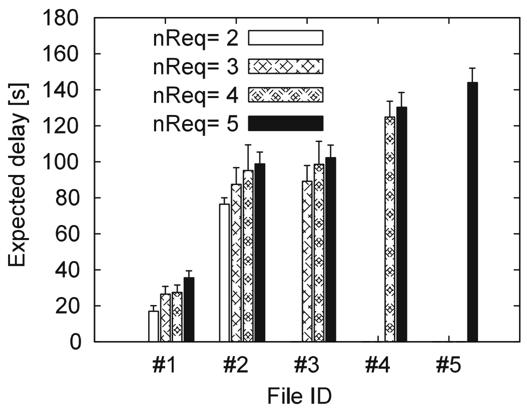


Fig. 8. Average transfer delay of full-duplex multiple requests.

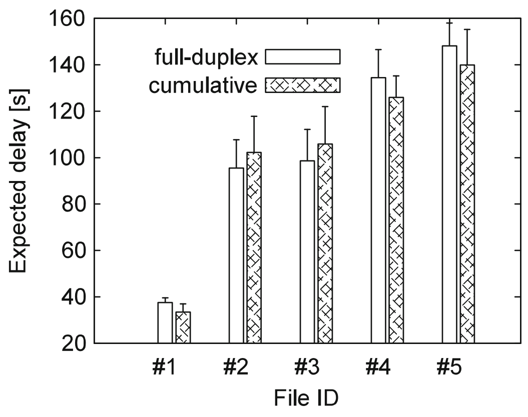


Fig. 9. Comparison of 2-nodes full-duplex request and 2-node single request of multiple files.

10 s. This scenario approximates the user behavior when she becomes incrementally aware of the surrounding context. Specifically, considering that contacts between moving users of the same community are asynchronous, and context information are exchanged asynchronously as well, user's requests will be delayed depending on her knowledge of the content availability. As shown in Fig. 10, the average transfer delay of delayed requested files, compared with those belonging to a cumulative request, is slightly improved.

The results presented in this section, even though they are related to a small-scale testbed, provide an indication to evaluate the critical time intervals related to the interactions of the context-aware content sharing service with a middleware platform aimed at the context management integrated with the Haggle architecture. However, since it is extremely important to analyze how the overall system can benefit from our middleware platform in more complex and more complete scenarios, we analyze in the next section simulation results that mainly address fundamental characteristics of opportunistic networks, like mobility and scalability.

7. Simulation results

In order to evaluate our content sharing application more extensively, we have developed a custom event-driven simulator written in C++.[7] In order to reduce the complexity of the system to a manageable level, we have assumed to have ideal wireless links with infinite bandwidth and negligible transmission delay. This is clearly unrealistic, but allows us to isolate the effect of context and social-awareness from networking effects such as congestion, transmission errors, etc., and provides a bound on the performance of the content sharing service, obtained in optimal settings.

We consider a reference scenario with three communities. In each community only a subset of all content items is available. The traveler nodes of the communities are the ones that, being the only ones that move outside their initial community, are in charge of carrying content items from one community to others. These users, as all the other users, have a limited buffer space and therefore the selection of the content items to be downloaded and prefetched is the crucial point in the content sharing process. Typically, the items in which the traveler users have a direct interest tend to receive a better service (recall that in the utility function in Eq. (1) there is a component exclusively dedicated to the interests of the local user). We decided to study the worst-case service received by less popular content, i.e., the case in which traveler users are interested in the most popular content. Apart from traveler users, users' interests are distributed within each community according to a Zipf's law with parameter $\alpha = 1$. Requests for content are generated by all users except the travelers according to a Poisson process with an average of 3 requests every 10 min. In the following, we focus on a tagged community and we evaluate the ability of the proposed content sharing policies to bring data to interested users. Results hold true regardless of the community chosen as reference. Simulations run for 50,000 s and exchanges of content items upon nodes' contacts start after an initial transitory phase required for the estimates of p_{ac} and p_{av} to reach their steady state. When applicable, results include their 95% confidence interval.

Nodes move according to the Home-cell Community-based Mobility Model (HCMM) (Boldrini and Passarella, 2010). This model is able to reproduce the aggregation into communities

[7] The code of the simulator is available upon request to the authors.

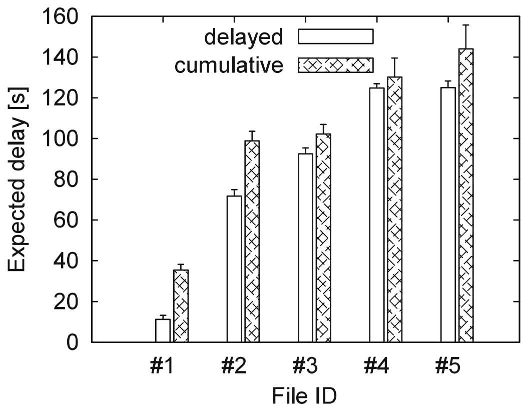


Fig. 10. 2-node full-duplex requests: multiple request vs single requests with a fixed interarrival time.

typical of the human users that make up an opportunistic network. The HCMM model starts by defining a social graph of network from the input parameters (e.g., number of nodes, number of communities, etc.). Specifically, each node is initially randomly assigned to one of the available communities, and all the nodes belonging to the same communities are fully connected with each other, but not with nodes belonging to a different community. Then, according to the Caveman model presented in Watts (1999), each of these links is rewired to a node belonging to different communities with a probability p_r, called *rewiring probability*. The rewiring probability is the knob to control how much the nodes that belong to different communities mix together when they move. In fact, in HCMM nodes move between communities, and nodes' movements are driven by social links between nodes. More specifically, the more the social links towards a community, the more likely the node will move towards that community. Thus, when the rewiring probability is low, the nodes are not much attracted to the outside of their community, thus they will rarely meet nodes belonging to a different community. When the rewiring probability is high, the opposite holds true. The HCMM has been shown in Boldrini and Passarella (2010) to be able to reproduce realistic features of human mobility, and thus it can be safely used in place of real mobility traces, when more flexibility in the definition of network scenario is needed.

All policies are evaluated in terms of the quality of service (QoS) perceived by the users and the resource consumption. The QoS is measured in terms of hit rate, system utility and fairness level. The hit rate is given by the number of successful requests divided by the number of overall requests. The system utility is computed as the sum of the content hit rate ($hitrate_i$) weighted with its popularity (pop_i), i.e., $SU = \sum_i pop_i \, hitrate_i$. The system utility goes from 0 to 1 (maximum utility). The fairness of each policy has been computed according to the traditional Jain's fairness index (using the hit rate as a measure of the service level obtained by each piece of content). Resource consumption has been measured in terms of the traffic generated in the network, i.e., the average number of data transmitted by all nodes during the simulation. This includes data exchanged for context creation, buffer state messages, request messages and downloaded content (Fig. 11).

The first scenario we consider comprises 15 nodes per community, moving according to the HCMM model, with the rewiring probability set to be 0.05. The rewiring probability is uniformly applied, thus the average number of social links across communities is the same for all nodes. Fig. 12 shows the hit rate experienced by all channels. All the policies presented in Section 4.2 guarantee approximately the same level of hit rate. This is an important result, as it shows that, although the rewiring probability is not particularly high, it is already sufficient to make the nodes, and thus the contents, circulate in the network.

In the next set of simulations we consider a less mixed scenario. Specifically, we consider the minimum level of connectivity that can guarantee a 100% hit rate, i.e., we assume that there are only two travelers in the network, one connecting the first and the second community, one connecting the first and the third community. Fig. 13 shows the system utility provided by each policy in this scenario. The policies that perform best are the Future and the Most Likely Next. The common strategy of these policies is that the traveler users do not cooperate or help disseminating messages within the community in which they are currently roaming, but proactively store data that might be of interest for the communities that they will visit in the future, as discussed in Section 4.1. The policies in which traveler users support data dissemination in the current community (MFV, Present and Uniform Social) see a decrease in the system utility of around 10%, which results from the synchronization between users in the same community. In fact, the users belonging to the same community "see" the same state of the network (in terms of object availability and access probability) and therefore, more or less simultaneously, take the same dropping/downloading decisions. But, if all users drop the same content items at about the same time, these items temporarily disappear from the local community and therefore all requests associated with them cannot be satisfied. The Greedy policy is, overall, the one performing worse. In fact, the Greedy policy can provide the nodes of the reference community only with those contents the traveler users are directly interested in, because they only download contents that are interesting for them. In this case, they are interested in the most popular items and thus they can satisfy the requests only of those nodes that are interested in the same items. The Uniform policy seems to work quite well from the system utility standpoint. However, in the following we show that this good performance is paid in terms of resource consumption.

Fig. 14 shows on a lin-log scale the traffic generated on average by the content sharing policies during simulations. The Uniform policy is the one that uses more resources and it shows a very high resource consumption. This is due to the fact that, with the Uniform policy, fetching decisions are taken uniformly at random. Therefore, also content items that are already well spread in a community can be exchanged, thus increasing the bandwidth demand. MFV, Present and Uniform Social are also quite demanding in terms of traffic due to the synchronization effect associated with these policies. In fact, after wrong dropping decisions, nodes must catch up with the content sharing process by disseminating contents that were already available before, but that have been dropped in the meanwhile due to synchronization.

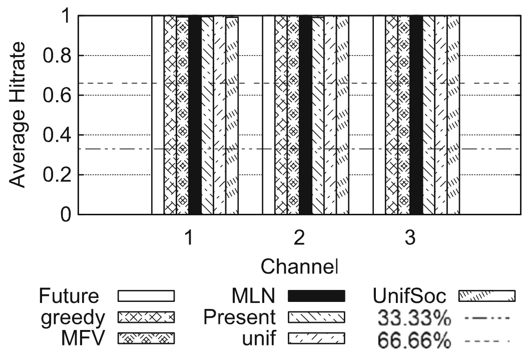


Fig. 11. Hit rate when the rewiring probability is set to 0.05.

The Greedy policy is extremely sparing with resources but this is paid in terms of system utility, as we have already shown. If we compare the results of system utility and the resource consumption, we find that the best trade-off between these two metrics is provided by the Future and MLN policies. We anticipate here that the Future and MLN policies will prove as the most effective and efficient solutions also with respect to metrics that we analyze in the remaining of this section.

With regard to the fairness provided by the content sharing policies (Fig. 15), all policies treat different contents equally except the Greedy policy. The Greedy policy is by definition unfair because under this policy users consider only their personal interests when taking downloading decisions. On the opposite, the Uniform policy is fair by definition, because it selects content uniformly at random. The social-based policies are fair because they also take into account the needs of less popular content, even if at a first stage they preferentially serve requests for more popular objects, and only once these requests have been satisfied, i.e., when the availability of these objects is high, less popular contents are downloaded.

So far we have assumed that the requests of nodes for content never expire. That is, the Content sharing service had an infinite time to deliver the content items requested by users. The previous analysis thus showed the performance bound of the different policies (in terms of hit rate) when infinite timeouts are allowed. Tables 2 and 3 show how the hit rate varies if we release this assumption. Specifically, they show for which timeout value a given percentile of hit rate can be achieved by the different policies (therefore, for a given percentile, the lower the timeout the better the policy performance). We consider the case of both the most popular and the least popular content. The Greedy policy completes the dissemination process very quickly (99% hit rate reached with a very short validity request) for the content (the most popular) that is interesting for the traveler user, but it cannot adapt to the needs of users other than the traveler user. In fact, if we consider contents that are not interesting for the traveler users (Table 3), the hit rate does not change as the timeout increases because the dissemination process is considered to be complete once the content that is interesting for the traveler user has been disseminated. Other content items are simply ignored and increasing the request validity does not help, as shown again in Table 3. All the other policies improve their performance when the timeout of the request is extended, but Future and MLN are much quicker than the other social policies in disseminating contents, both for popular and less popular content. Therefore, the Future and MLN policies provide the best service in terms of percentage of reached users for shorter validity timeouts. Note also that in Tables 2 and 3 there are cases in which a certain level of hit rate is never reached (referred in tables as $\geq 50{,}000$, where 50,000 s is the duration of our simulation). The duration of the simulation has been chosen to be large enough with respect to the mobility settings of our scenario, but obviously it cannot be considered infinite. Thus, results in Tables 2 and 3 are not an evidence for the inability of some policies to reach certain levels of hit rate, however they are a good indication of this behavior.

Summarizing the results obtained so far, in the scenario that we have considered the best trade-off between QoS perceived by users and resource consumption is provided by both the Future and MLN policies. Their advantage over the other social policies is that they exempt traveler users from cooperating with the nodes of the local community and allow them to proactively download content that will be interesting for the nodes that they will meet in the future. For what concerns non-social policies, the Uniform

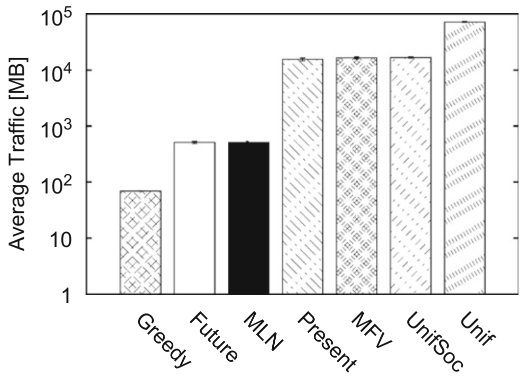


Fig. 13. Resource consumption.

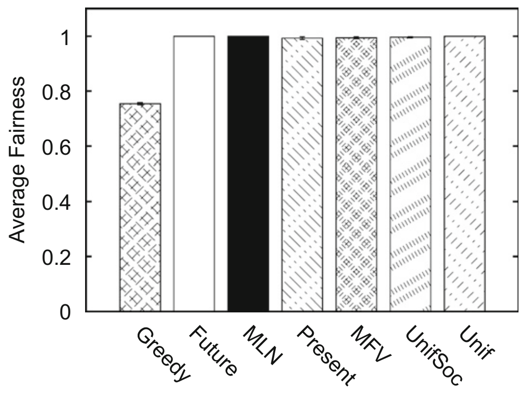


Fig. 14. Fairness level.

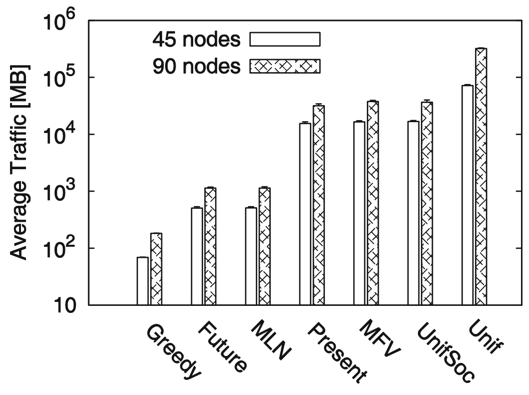


Fig. 15. Resource consumption (45 vs 90 nodes).

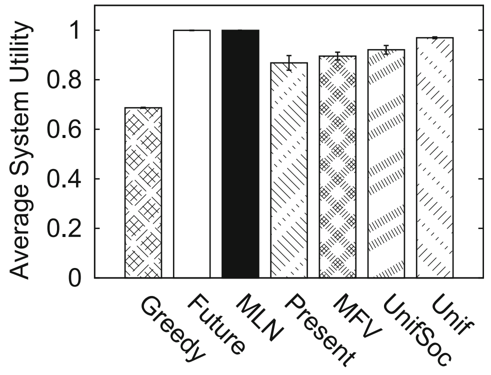


Fig. 12. System utility.

policy wastes a lot of resources and this may not be practical in a real network. Instead, the Greedy policy is extremely unfair and can excessively penalize less popular content.

The next set of simulations focuses on the performance of the content dissemination schemes when the number of the nodes in the network is increased. Specifically, in the following we compare the previous scenario with 45 nodes with the case of 90 nodes, divided equally into the three communities. All other configuration parameters remain the same as above. Tables 4 and 5 show how the hit rate varies for the most and the least popular channel when we reduce the request validity. For ease of readability, we have removed MLN, Present, and Uniform Social from the table. In fact, MLN has shown to be approximately equivalent to the Future scheme in this scenario, and, similarly, Present and Uniform Social can be reasonably approximated by the MFV scheme. In the case of the most popular channel, there are no significant changes in the hit rate experienced. In fact, with both 45 and 90 nodes, only the Greedy and the Future policies are able to reach 99% of hit rate. For the most popular channel the dissemination process accelerates as the number of nodes increases in all the non-social cases. Social-aware policies seem to slightly suffer from the increase in the number of nodes. Specifically, the performance of the schemes that suffer from the synchronization effect (here represented by MFV) requires a request validity around three times larger than the request validity of the 45 nodes case in order to reach the same level of hit rate level. The reason is that, with more nodes per community, the synchronization effect is stronger than before, because the encounters between nodes are more frequent. The difference in the performance of the Future scheme with 45 and 90 nodes is very small, and it is satisfactorily made up for by the hit rate of the Future policy for the least popular content. In fact, as shown in Table 5, the only scheme that can guarantee a 99% hit rate is the Future scheme. With the Greedy scheme, traveler nodes only retrieve content that is interesting for them, and the least popular channel (that is not the one to which traveler nodes are subscribed) is excessively penalized. As a general comment, please note that content that is not very popular benefits from an increase in the number of nodes in the network, which, as a matter of fact, results in a decrease in the request validity needed to obtain a certain level of hit rate.

Fig. 15 compares resource consumption in the case of 45 and 90 nodes. The variation in the amount of resources consumed is the smallest (around 120%) for all the social policies, while it is approximately 165% for the Greedy scheme and 350% for the Uniform policy. Besides confirming as the scheme that consumes more resources, the Uniform policy becomes also the scheme whose performance significantly worsens with an increased number of nodes. This set of results confirms that, also in the case of an increased number of nodes, the Future policy (and thus the MLN policy, which performed in much the same way in our simulations) is the one that guarantees a good QoS (small delay, fair treatment of all contents) while at the same time limiting resource consumption.

We conclude this evaluation by showing that the Future and Most Likely Next policies are not equivalent in all scenarios as they were in the scenario studied so far. By definition, when taking downloading/dropping decisions, the Most Likely Next policy takes into account only those communities that are at "1-hop social distance" from the current one (see Section 4.1), i.e., those communities that the traveler nodes will visit next. In fact, the MLN policy computes the social weights by conditioning on the current community of the traveler user. As an example, if a user goes to the gym after work, but only after having dropped suitcase and work-related stuff at home, the MLN policy will not be able to fetch content from the Work community that might be of interest for the members of the Gym community, because Work and Gym are not visited one after the other. On the bright side, however, this strategy allows the user to exploit its buffer space to download content interesting for the members of the communities that it will visit in the very next future. The Future policy, as the name suggests, considers all the communities that can be visited by the traveler users, not only the next one. In order to highlight these different behaviors, we run additional simulations with a slightly different scenario. We considered three communities (referred to as C1, C2, C3) and one traveler node that visits these communities according to the following rule: communities C2 and C3 can be reached only from community C1. Thus, C2 and

Table 2
Request validity (measured in seconds) and expected hit rate for the most popular content.

Hit rate	Future	Greedy	MFV	MLN	Present	Unif	UnifSoc
50-th percentile	0	0	1.1904	0	1.4189	0.0459	0.6191
75-th percentile	0.0053	0	6.6941	0.0048	7.9033	0.3222	4.4902
99-th percentile	18.345	0.6153	$\geq$50,000	18.345	$\geq$50,000	$\geq$50,000	$\geq$50,000

Table 3
Request validity (measured in seconds) and expected hit rate for the least popular content.

Hit rate	Future	Greedy	MFV	MLN	Present	Unif	UnifSoc
50-th percentile	0	$\geq$50,000	0.0201	0	0.0222	0.0333	0.0252
75-th percentile	0	$\geq$50,000	3.3556	0	7.7419	0.256	3.1328
99-th percentile	25.7833	$\geq$50,000	$\geq$50,000	25.7833	$\geq$50,000	$\geq$50,000	$\geq$50,000

Table 4
Request validity (measured in seconds) and hit rate for the most popular content (45 vs 90 nodes).

Hit rate	Greedy 45	Greedy 90	Future 45	Future 90	MFV 45	MFV 90	Unif 45	Unif 90
50-th percentile	0	0	0	0	1.1904	3.2405	0.0459	0.0199
75-th percentile	0	0	0	0053 0	6.6941	20.2406	0.3222	0.0984
99-th percentile	0.61526	0.0357	18.345	21.7381	$\geq$50,000	$\geq$50,000	$\geq$50,000	$\geq$50,000

Table 5
Request validity (measured in seconds) and hit rate for the least popular content (45 vs 90 nodes).

Hit rate	Greedy 45	Greedy 90	Future 45	Future 90	MFV 45	MFV 90	Unif 45	Unif 90
50-th percentile	$\geq$ 50,000	$\geq$ 50,000	0	0	0.0201	0	0.0333	0.0163
75-th percentile	$\geq$ 50,000	$\geq$ 50,000	0	0	3.3556	0.0723	0.256	0.0892
99-th percentile	$\geq$ 50,000	$\geq$ 50,000	25.7833	10.4624	$\geq$ 50,000	$\geq$ 50,000	$\geq$ 50,000	$\geq$ 50,000

Table 6
Request validity (measured in seconds) and hit rate for the content at 1-hop distance.

Hit rate	Future	MLN
50-th percentile	0	0
75-th percentile	0	0
99-th percentile	1.4768	2.72937

Table 7
Request validity (measured in seconds) and hit rate for the content at 2-hops distance.

Hit rate	Future	MLN
50-th percentile	0	0
75-th percentile	0	$\geq$ 50,000
99-th percentile	32.8666	$\geq$ 50,000

C3 can be considered at a 2-hop distance from each other if we measure hops in terms of communities to be visited when going from C2 to C3. Using this definition, C1 and C2, and C1 and C3, are at a 1-hop distance. We assume that contents are initially allocated evenly to each community (i.e., one third each). From what we have said so far, we expect that MLN will not be able to download contents assigned to community C2 and carry them to community C3, because these two communities are not at 1-hop distance. Tables 6 and 7 confirm the expected results: while the MLN policy is able to quickly deliver content among communities at 1-hop from each other, one third of the content residing in communities at 2-hops from each other can never be delivered by the MLN policy, regardless of the settings of the request validity, while 1-hop content is correctly disseminated. On the contrary, the Future policy is able to reach 99% hit rate in both cases, if requests remain valid for enough time.

From our simulations, Future and MLN has both emerged as very effective and efficient policies. However, we believe that in a real environment, restricting content sharing to communities that are at 1-hop distance can be too restrictive. Thus, we propose the Future policy as the policy that can provide very good QoS to users, save resources in terms of traffic generated across the network, and exploit multi-hop social paths between communities.

8. Conclusions

In this paper we have proposed a context- and social-aware middleware that autonomically learns context information and users' sociality, and makes this information available to all the components of the network architecture, from the forwarding level up to the application level. Our middleware has been integrated in the Haggle architecture, of which it inherits the layer-less spirit and the data-centric approach. Due to the modularity of Haggle architecture, our middleware has been implemented as an additional manager (i.e., Context manager) aimed at managing and disseminating context information to improve network communications, protocols, and services. Then, on top of this architecture, we have designed and developed a context-aware content sharing service that exploits the interaction with the Context manager inside Haggle to exploit context information for healing network partitions among different communities and intermittent connectivity conditions. To this aim, we have proposed a variety of policies to evaluate the impact of users' behavior on data dissemination among separate communities, ranging from completely social- and context-oblivious to fully social- and context-aware. The enhanced architecture and the content-sharing service have been validated on a small-scale using a real testbed, while simulations have been used to provide a larger scale evaluation, including realistic human mobility.

Our evaluation has shown that social-aware sharing policies are the most effective and efficient strategies for content dissemination in opportunistic networks, both from the standpoint of the QoS perceived by users and the resource consumption. These policies cannot be implemented without a middleware that takes care of the process of collecting and managing context and social information. Therefore, the proposed context- and social-aware middleware becomes the enabling component for efficient communications in opportunistic networks.

Acknowledgement

This work was partially funded by the IST program of the European Commission under the HAGGLE (027918) and SOCIALNETS (217141) FET Projects.

References

Balamash A, Krunz M. An overview of web caching replacement algorithms. IEEE Communications Surveys & Tutorials 2004;6(2):44–56.

Balasubramanian A, Levine B, Venkataramani A. DTN routing as a resource allocation problem. Applications, Technologies, Architectures, and Protocols for Computer Communication 2007;37:4.

Bellavista P, Corradi A. The handbook of mobile middleware. Auerbach Publications; 2007.

Boldrini C, Conti M, Passarella A. ContentPlace: social-aware data dissemination in opportunistic networks. In: Proceedings of ACM MSWiM 2008. Vancouver, Canada;2008a. p. 203–10.

Boldrini C, Conti M, Passarella A. Exploiting users social relations to forward data in opportunistic networks: the HiBOp solution. Pervasive and Mobile Computing 2008b;4(5):633–57.

Boldrini C, Passarella A. HCMM: modelling spatial and temporal properties of human mobility driven by users' social relationships. Computer Communications 2010; in press. doi: 10.1016/j.comcom.2010.01.013.

Campbell A, Eisenman S, Lane N, Miluzzo E, Peterson R, Hong L, et al. The rise of people-centric sensing. IEEE Internet Computing 2008;12(4):12–21.

Conti M, Delmastro F, Passarella A. Social-aware content sharing in opportunistic networks. In: Proceedings of IEEE SECON 2009. Rome, Italy; 2009. p. 1–3.

Conti M, Giordano S. Multihop ad hoc networking: the reality. IEEE Communications Magazine 2007;45(4):88–95.

Conti M, Kumar M. Opportunities in opportunistic computing. Computer 2010;43(1):42–50.

Costa P, Mascolo C, Musolesi M, Picco GP. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. IEEE Journal on Selected Areas in Communications 2008;26(5):748–60.

Daly EM, Haahr M. Social network analysis for routing in disconnected delay-tolerant MANETs. In: Proceedings of the 8th international symposium on mobile ad hoc networking & computing; 2007. p. 32–40.

Danon L, Díaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment 2005:P09008.

Dey AK. Understanding and using context. Personal and Ubiquitous Computing 2001;5(1):4–7.

Fall K. A delay-tolerant network architecture for challenged internets. In: Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications. Karlsruhe, Germany; 2003. p. 27–34. doi:http://doi.acm.org/10.1145/863955.863960.

Gao W, Li Q, Zhao B, Cao G. Multicasting in delay tolerant networks: a social network perspective. In: Proceedings of the 10th international symposium on mobile ad hoc networking & computing; 2009. p. 299–308.

Henricksen K, Indulska J, Rakotonirainy A. Modeling context information in pervasive computing systems. Lecture Notes in Computer Science 2002: 167–80.

Henricksen K, Indulska J. Developing context-aware pervasive computing applications: models and approach. Pervasive and Mobile Computing 2006;2(1):37–64.

Hui P, Crowcroft J, Yoneki E. Bubble rap: social-based forwarding in delay tolerant networks. In: Proceedings of the 9th ACM international symposium on mobile ad hoc networking and computing; 2008. p. 241–50.

Hui P, Crowcroft J. How small labels create big improvements. In: Proceedings of the 5th IEEE international conference on pervasive computing and communications workshops; 2007. p. 65–70.

Hui P, Yoneki E, Chan SY, Crowcroft J. Distributed community detection in delay tolerant networks. In: Proceedings of the 2nd ACM/IEEE international workshop on mobility in the evolving internet architecture; 2007. p. 1–8.

Lee U, Magistretti E, Gerla M, Bellavista P, Corradi A. Dissemination and harvesting of urban data using vehicular sensor platforms. IEEE Transactions on Vehicular Technology 2009;58(2):882–901.

Lenders V, Karlsson G, May M. Wireless ad hoc podcasting. In: Proceedings of the 4th annual IEEE communications society conference on Sensor, Mesh and Ad Hoc communications and networks; 2007. p. 273–83.

Lindgren A, Doria A, Schelen O. Probabilistic routing in intermittently connected networks. In: ACM SIGMOBILE mobile computing and communications review; 2003. p. 19–20.

Milgram S. The small world problem. Psychology today 1967;2(1):60–7.

Miluzzo E, Lane ND, Fodor K, Peterson R, Lu H, Musolesi M, et al. Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application. In: Proceedings of the 6th ACM conference on Embedded network sensor systems; 2008. p. 337–50.

Newman ME. The structure and function of complex networks. Arxiv preprint cond-mat/0303516 2003.

Newman ME. Detecting community structure in networks. The European Physical Journal B-Condensed Matter and Complex Systems 2004;38(2):321–30.

Roccetti M, Gerla M, Palazzi C, Ferretti S, Pau G. First responders' crystal ball: how to scry the emergency from a remote vehicle. In: Proceedings of the IEEE international performance, computing, and communications conference, IPCCC; 2007. p. 556–61.

Schmidt A. Ontology-based user context management: the challenges of imperfection and dynamics. In: Proceedings of the international conference on Ontologies, Databases and Applications of Semantics (ODBASE 2006), On The Move federated conferences (OTM 2006). Montpellier; 2006. p. 995–1011.

Shikfa A, Önen M, Molva R. Privacy in context-based and epidemic forwarding. In: Proceedings of IEEE AOC 2009; 2009. p. 1–7.

Silvis J, Niemeier D, D'Souza R. Social networks and travel behavior: report from an integrated travel diary. In: Proceedings of the 11th international conference on travel behaviour research. Kyoto; 2006.

Sollazzo G, Musolesi M, Mascolo C. TACO-DTN: a time-aware content-based dissemination system for delay tolerant networks. In: Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking; 2007. p. 83–90.

Spyropoulos T, Psounis K, Raghavendra CS. Spray and focus: efficient mobility assisted routing for heterogeneous and correlated mobility. In: Proceedings of the 5th IEEE international conference on pervasive computing and communications workshops; 2007. p. 79–85.

Spyropoulos T, Psounis K, Raghavendra CS. Efficient routing in intermittently connected mobile networks: the multiple-copy case. IEEE/ACM Transactions on Networking (TON) 2008;16(1).

Strang T, Linnhoff-Popien C. A context modeling survey. In: Workshop on advanced context modelling, reasoning and management as part of UbiComp, 2004.

Tanenbaum AS, Van Steen M. Distributed systems principles and paradigms. Prentice-Hall; 2002.

Vahdat A, Becker D. Epidemic routing for partially connected ad hoc networks. Technical report CS-2000-06, 2000.

Van Jacobson DK, Briggs TM, Braynard RL. Networking named content. In: Proceedings of the 5th ACM international conference on emerging networking experiments and technologies (CoNEXT 2009). Rome, Italy; 2009. p. 1–12.

Wang Y, Lin F, Wu H. Cross-layer protocol design for delay/fault-tolerant mobile sensor networks. ACM SIGMOBILE Mobile Computing and Communications Review 2007;11:2.

Watts DJ. Small worlds: the dynamics of networks between order and randomness. Princeton: Princeton University Press; 1999.

Yoneki E, Hui P, Chan SY, Crowcroft J. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In: Proceedings of the 10th ACM symposium on modeling, analysis, and simulation of wireless and mobile systems; 2007. p. 225–34.

Zhao W, Ammar M, Zegura E. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In: Proceedings of the 5th international symposium on mobile ad hoc networking & computing; 2004. p. 187–98.