SERIES OF PUBLICATIONS A
REPORT A-2007-5

A Decentralized Session Management Framework for Heterogeneous Ad-Hoc and Fixed Networks

Simone Leggio

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XIV University Main Building, on November 17th, 2007, at 10 o' clock.

UNIVERSITY OF HELSINKI
FINLAND

Contact information

Postal address:
Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: postmaster@cs.Helsinki.FI (Internet)

URL: http://www.cs.Helsinki.FI/

Telephone: +358 9 1911

Telefax: +358 9 191 51120

ISSN 1238-8645
ISBN 978-952-10-4221-8 (paperback)
ISBN 978-952-10-4222-5 (PDF)
Computing Reviews (1998) Classification: C.2.0,C.2.2,C.2.4
Helsinki 2007
Helsinki University Printing House

A Decentralized Session Management Framework for Heterogeneous Ad-Hoc and Fixed Networks

Simone Leggio

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
simone.leggio@cs.helsinki.fi
http://www.cs.helsinki.fi/u/leggio/

PhD Thesis, Series of Publications A, Report A-2007-5
Helsinki, 2007, 230 pages
ISSN 1238-8645
ISBN 978-952-10-4221-8 (paperback)
ISBN 978-952-10-4222-5 (PDF)

Abstract

Wireless technologies are continuously evolving. Second generation cellular networks have gained worldwide acceptance. Wireless LANs are commonly deployed in corporations or university campuses, and their diffusion in public hotspots is growing. Third generation cellular systems are yet to affirm everywhere; still, there is an impressive amount of research ongoing for deploying beyond 3G systems. These new wireless technologies combine the characteristics of WLAN-based and cellular networks to provide increased bandwidth. The common direction where all the efforts in wireless technologies are headed is towards an IP-based communication.

Telephony services have been the killer application for cellular systems; their evolution to packet-switched networks is a natural path. Effective IP telephony signaling protocols, such as the Session Initiation Protocol (SIP) and the H 323 protocol are needed to establish IP-based telephony sessions. However, IP telephony is just one service example of IP-based communication. IP-based multimedia sessions are expected to become popular and offer a wider range of communication capabilities than pure telephony.

In order to conjoin the advances of the future wireless technologies with the potential of IP-based multimedia communication, the next step would be to obtain ubiquitous communication capabilities. According to this vision, people must be able to communicate also when no support from an infrastructured network is

available, needed or desired. In order to achieve ubiquitous communication, end devices must integrate all the capabilities necessary for IP-based distributed and decentralized communication. Such capabilities are currently missing. For example, it is not possible to utilize native IP telephony signaling protocols in a totally decentralized way.

This dissertation presents a solution for deploying the SIP protocol in a decentralized fashion without support of infrastructure servers. The proposed solution is mainly designed to fit the needs of decentralized mobile environments, and can be applied to small scale ad-hoc networks or also bigger networks with hundreds of nodes. A framework allowing discovery of SIP users in ad-hoc networks and the establishment of SIP sessions among them, in a fully distributed and secure way, is described and evaluated. Security support allows ad-hoc users to authenticate the sender of a message, and to verify the integrity of a received message.

The distributed session management framework has been extended in order to achieve interoperability with the Internet, and the native Internet applications. With limited extensions to the SIP protocol, we have designed and experimentally validated a SIP gateway allowing SIP signaling between ad-hoc networks with private addressing space and native SIP applications in the Internet. The design is completed by an application level relay that permits instant messaging sessions to be established in heterogeneous environments. The resulting framework constitutes a flexible and effective approach for the pervasive deployment of real time applications.

Computing Reviews (1998) Categories and Subject Descriptors:
C.2.0 Computer-Communication Networks: General
C.2.2 Computer-Communication Networks: Network Protocols
C.2.4 Computer-Communication Networks: Distributed Systems
C.2.4 Computer-Communication Networks: Internetworking

General Terms:
Design, Experimentation, Algorithms

Additional Key Words and Phrases:
Session Initiation Protocol, Session Management, Internet Signaling, Real Time Communication, Mobile Ad-Hoc Networks, Decentralized and Distributed Systems, Heterogeneous Networks Interworking, Peer-to-Peer Communication

Acknowledgements

Helsinki, May 19th, 2007

I still remember that sunny spring 2001 day in Catania, when I knocked the door of Prof. Sergio Palazzo's office, having just won an Erasmus scholarship. The discussion, which I have repeated countless times to people asking me the reasons for my being in Finland, went more or less like this:

- Prof, I have won an Erasmus scholarship to prepare the M.Sc. thesis somewhere in Europe. Here is the list of nations where I could go, what do you suggest me?-

I was expecting (and maybe hoping) that he would send me to England, so to improve my English, but his reply was somewhat surprising:

- I think you should go to Finland. It's a very good place for Telecommunications, with Nokia and the rest, and I have a contact there, Prof. Kimmo Raatikainen, who will be glad to help you.-

- Prof, but Finland is cold, are you sure it's a good place? -

- Yes, I'm sure. Go *and you won't repent it.* -

Well, 6 years after that discussion I'm still in Finland, for a reason. I wish to thank prof. Palazzo who gave me the possibility to turn my life with a 180 degrees angle, probably not imagining himself the changes he was going to bring about in my life.

My deepest gratitude goes to prof. Kimmo Raatikainen, who accepted me as a M. Sc. student six years ago, agreed that I could continue my studies as Ph.D. candidate with him and highly contributed to my growth as a person, a student and a professional. Thank you, heartily, with another professor maybe I would not be where I am now.

There is a person that has always been on my side throughout my academic career so far, and that is Prof. Jukka Manner. Prof. Manner has been a continuos source of suggestions, comments and insights for me, since the times of my preparing the M.Sc. thesis. His help during these years has simply been unvaluable, and I could not have done what I have done without his support. Thank you again for 5 years of great cooperation, it has been a pleasure to work together.

I wish to thank all the people in the department who helped me through the long path towards the Doctorate, particularly Markku Kojo, who has been my mentor in my early times at the department, and contributed to my academic growth. Marina Kurten deserves my gratitude as she made my english become less *italian* revising the language of my papers and thesises throughout these years.

The bulk of the dissertation is based on a national Finnish project, SESSI. I wish to thank all my colleagues in that project for two years of productive work that have led to really considerable results. I thank Prof. Antti Ylä-Jääski and Prof. Tommi Mikkonen who contributed to steer such a successfull project. It also thanks to the high quality of work produced by SESSI that I am here writing this text.

I spent marvellous moments together with Hugo Miranda, a researcher from the University of Lisbon, Portugal, who is probably writing the acknowledgments for his Ph.D. dissertation while I am typing these very lines! We were more than colleagues in the few moments we were able to spend together, and I can say we really achieved something by being able to work at a distance. Thank you Hugo for your having introduced me to the world of distributed systems and again congratulations for the idea you had when you first thought of PCache!

I am grateful to Prof. Luis Rodrigues, who accepted my visit to the University of Lisbon, giving me the possibility of finalizing the PCache algorithm. I also acknowledge the economic support received from the MiNEMA program, which made my visit possible.

Five years abroad cannot be easy for anyone. One person has shared with me practically everything about my experience: he was also in Prof. Palazzo's room that day, he has done the M.Sc thesis at the same time with me and worked on his Ph.D. in parallel with me. Thank you to Davide Astuti for his support as a friend, a brother maybe, during these years far from our Sicily. I have gone through difficult moments, and I could overcome them also because I knew I could count on a friend like him. I hope now that despite our paths are diverging we can still continue to share the same emotions and experiences as we have done so far.

With Giovanni Costa, the third person in that room that day, unfortunately this was not possible: after he left Finland my fears became true. We could no longer be "united" as we had been since the university times. I will never forget those times, Giovanni, no matter where the destiny will decide to take me.

During these years I have had the luck to have on my side the best person I could have ever possibly desired. Infinite gratitude, paljon kiitoksia, to my love, Anne. When I met her, a few days after I decided to continue as a Ph.D. (maybe it was a gift for having had the courage of prolonging my stay here) student, I could have never imagined that I would have been so lucky to find her. Her simple presence on my side gives me hope for a better tomorrow and cheers me up when

I feel down.

I want to conclude this section in the same way as I did my M.Sc. thesis:

Last but not least, thanks to my family, even if physically far, I always felt them as if they were in Finland with me.

Grazie, tante grazie, mamma, papa' e Oriana, per la vostra comprensione e pazienza nel vedermi qui lontano. So che e' difficile vivere cosi' lontani l'uno con gli altri, ma sappiate che sempre vi penso e vi voglio bene, anche se purtroppo il destino ha avuto in serbo una tale sorpresa per me e per voi. Ma non siate tristi, io mi ripeto sempre questa frase: - Pero', a 29 anni ho fatto qualcosa di buono...- Questo deve essere quello che mi spinge a fare ancora meglio e a sopportare la mia lontananza da voi e da Ragusa. In fondo, sappiate che quello che faccio lo faccio perche' mi piace e questo mi aiuta a superare ogni traversia.

Contents

Chapter 1

Introduction

Wireless technologies have reached an advanced state of development. Improvements in the medium access techniques and in the capabilities of terminals, in terms of battery duration, processing power, and amount of storage space, have given birth to a large set of new applications, services, terminals and research issues and challenges. Users, also the less technologically orientated ones, are learning now to expect more than just voice call capabilities from their wireless devices; the steady growth in the sales of smartphones is an example of such a trend.

The last generation of wireless devices has built-in capabilities that make them digital assistants, mobile phones, music players or even portable TV sets at the same time. The growth in terminal capabilities has been partly motivated by the widespread development of high-speed technologies for wireless communication. Corporate WLANs now customarily accompany the wired Ethernet counterpart, and ensure continuously increasing data speeds. Service providers offer WLAN access in hot-spots, like airports or big malls, to their customers. Even though currently they are used mainly for business reasons and for a limited set of services, like reading e-mail, WLANs deployed in hot-spots are expected to become very popular in the next few years, and similarly the set of available services is expected to grow.

Third generation networks have become popular, promising fast data rates whenever and wherever needed. In other words, the availability of high data speed through handheld devices is now becoming reality. Higher bandwidth availability allows the design of more complex services and applications that fully utilize such a commodity. Data traffic, possibly carried over a packet-switched network such as the Internet, is predicted to become dominant; packet-switched networks are foreseen to substitute the older circuit-switched networks even for pure voice traffic. Indeed, the core networks of the most recent cellular technologies, like GPRS or the 3G Internet Multimedia Subsystem, IMS, are IP-based.

The key factor that glues everything together is an increased communication capability. Users want to exploit it and want to communicate and interact with each other, with a much broader set of possibilities than solely the voice: it is a general belief that video calls or instant messaging sessions from mobile handsets could become killer applications, and take the role that is currently played by the Short Message System (SMS) messaging.

The research challenges are to develop the means for enabling such communication capabilities. From the networking point of view, network engineers should build protocols allowing communication and exchange of data in real time among users. These protocols must guarantee the quality of the provided services. From the application point of view, application developers should build as compatible software as possible, for enabling communication among devices of different brands and types, each probably using a different communication medium. The necessary adaptation needed for handling this variety is a great challenge as well.

The final step is a total integration between the network and the device, and between various networks. While 3G networks are not yet fully widespread, already researchers are studying 4G networks, where WLAN or other high-speed technologies will be embodied with cellular networks. The completion of the picture will be a pervasive, ubiquitous wireless world [135].

1.1 A Vision for the Future of Mobile Computing

The feeling that the mobile computing world was on the threshold of a revolution has been shared by several scientists, and for a long time already. Mark Weiser gave a vision on the computer of the 21st century [135] in 1991. His article has created the basis for extensive work, still continuing nowadays, where several of the problems he outlined are yet unresolved. Mark Weiser's paradigm is referred to as *ubiquitous* or *pervasive* computing.

The core of Weiser's idea is that computing facilities should be transparently integrated with the everyday environment surrounding the user, so that the user is unaware of the presence of computing devices that pervade his/her environment. People should feel with computing devices the same way as they would feel with electric switches. His vision has not, 15 years later, yet come true. It was for that period far too challenging, and it currently still is. On the other hand, using a computing device is not as easy as switching the lights on and off.

Kurt Geihs [38] provides an insight on the challenges that middleware researchers and programmers must face for designing next-generation middleware. He stresses that a middleware layer is necessary for performing adaptation operations that the variety of devices that constitute the future networking and computing environment entails.

Adaptation is also the magic word for Randy Katz [58]. Implementors, besides focusing on producing newer and fancier devices and applications, should also address interoperability and adaptation needs that the big variety of devices in a mobile network calls for.

Leonard Kleinrock [62] is more concerned with the pervasiveness of the Internet. As the main designer of the current Internet, Kleinrock wished, that Internet access would be available everywhere and to everyone, as one would do when plugging an electric device to a socket. This is not true, and the realization of Kleinrock's vision is a challenge for the future. Authentication of users and authorization of access rights are still a big unresolved problem for the current Internet.

Mahadev Satyanarayanan's vision on pervasive computing [114], also based on Weiser's, opens interesting scenarios, which we deem important for our personal vision of the future of mobile computing. Due to the pervasiveness of computing facilities, which basically make services and computing capabilities available everywhere, he claims that means should be given to users for discovering the services available in the particular environment where they happen to be. Such possibility allows to fully exploit the pervasiveness of services in the surrounding computing environment.

Kimmo Raatikainen builds on the previous work and provides his vision and challenges for future mobile computing [95]. Raatikainen envisages a world where the users have control not of a single end device, but of a whole set of devices, in other words, reconfigurable end-user systems. While context awareness plays an important role, users still must control their personal system. Should the devices be too much in control of users' habits, the trust in the device and its context-aware capabilities would get lost. Users will mainly use their end systems for communication; they do not want to feel isolated, but live in a (virtual) community, with the possibilities of easy interaction with their professional or private contacts.

Our central proposition is that communication is the key of future mobile computing. We envisage a pervasive communication environment, where people are able to communicate, with other people or also with machines, providing intelligent, value-added services, as transparently and as seamlessly as possible.

We imagine people walking by in a mall, and contacting friends who happen to be there shopping at the same time, or why not, also strangers with whom they share the same interests, to discuss about a new dress to buy. Users do not need to explicitly call a remote friend, nor to pay for the connection. Communication may happen among peers; this is a distinctive factor compared to the present situation. Today, basically all communication traffic is relayed through the infrastructure of a network operator. To enable pervasive communication, when and where it

is needed, it is not possible to rely on the support from the network. The end devices must be capable of supporting the communication on their own, without involvement of network infrastructure.

We imagine a traveler waiting in an airport lounge for his plane to leave, taking his PDA and engaging in a chess game with a nearby traveler. The PDAs have the capabilities to instantiate the connection and communicate directly, and without involvement of existing network infrastructure. Additionally, an operator-provided access point in the range advertises the possibility of toll-based access to the Internet. The traveler may decide to stop the game and read his e-mail using the connection provided by the operator.

In a huge conference hall, two work colleagues receive notifications of each other's presence on their devices and engage in a secured chat session. No one else is aware of the content of exchanged messages, as communication is encrypted and happens directly between the laptops of the two colleagues. An ad-hoc communication framework can also be useful in a crisis situation, where a team of rescuers is deployed in an area affected by a natural calamity. In such a case, the support from the infrastructure may not be available, so establishing peer-to-peer communication channels over ad-hoc networks among the rescue team members is a feasible communication method.

Communication capabilities must therefore pervade the user's environment. The only way to achieve total pervasiveness is to equip the user end devices with the facilities for communicating on their own.

1.2 Motivation

The vision described entails a strong necessity for managing communication among peers, in a wide range of networking environments. The role of the network operator as the only provider of communication means to connected devices will fade. Instead, the rise of ad-hoc communities, where users connect their devices directly with each other, without resorting to a pre-existing network infrastructure will make infrastructureless peer-to-peer communication become as popular.

A new role for the operator can be defined, e.g., as provider of new type of services to an ad-hoc community. For example, operators can provide ad-hoc users with access to an external network, such as Internet or 3G, or access to a printing service. If users have to find computing capabilities everywhere, it is not reasonable nor practical to assume the infrastructure will always be present. Nor it will be necessary to utilize the operator resources if a direct communication between two devices is achievable. If nearby users can connect directly with each other, the network can be made more scalable, and there are more resources available for those who need to communicate with remote parties and must resort to the

network support. Sometimes it may even be beneficial to use ad-hoc direct communication rather than rely on the infrastructure, as in the case of a VoIP session with a nearby person. If e.g. GPRS is used, the time needed for relaying all the communication through GPRS nodes in the backbone network, possibly far away from the user's location, may highly exceed the delay of direct peer-to-peer data transfer.

Communication among users must rely on a decentralized and distributed approach, to cope with situations where the support from the network is not available or convenient. The peer-to-peer networking scheme must seamlessly accompany and interact with the traditional client-server approach. The preferred way of communication will be IP-based, as it is expected that a growing part of communication data will be carried over IP networks, while the older circuit-switched telephony networks will play a secondary role, even for voice communication.

In order to efficiently set up a communication between two or more parties, proper signaling is needed to negotiate the parameters and characteristics of the media(s) used for communication. The operation of signaling for the establishment of media sessions is also referred to as session management. Nevertheless, currently all the protocols for session management require the presence of a pre-existing network infrastructure, as they leave most of the operations to centralized entities, or servers. To support the proposed vision and needs of future communication, it is necessary to deploy session management schemes in a distributed way. In this way, IP-based media exchanges can also be possible in ad-hoc communities, or in general, in environments where no support from the infrastructure is available or needed.

Moreover, to fully enable pervasive communication, it is necessary that applications and services modified for use in ad-hoc networks and in a peer-to-peer fashion, are interoperable with existing correspondent Internet applications. The fact that the users can use the same application in ad-hoc networks as in the Internet, rather than having to use two different ones is a much higher guarantee for success for the pervasive communication concept. Further, it is also important that interoperability is guaranteed even simultaneously, when possible, and users can establish communication sessions with peers in the proximity or with contacts remotely located somewhere and accessible through the Internet.

It is thus necessary not only to properly modify existing applications so that they can be utilized in a peer-to-peer fashion, but also that the design is flexible enough to maintain interoperability with existing services and protocols. This dissertation shows a solution that covers both of the aspects described above.

1.3 Overview of the Approach

There are two major signaling protocols for IP-based media exchanges; the first, the Session Initiation Protocol (SIP) [106] has been standardized by the Internet Engineering Task Force (IETF) [143]. The other signaling protocol is actually a framework and a suite of protocols, collectively referred to as the H 323 protocol suite [47], and has been developed by ITU-T [144].

We have chosen SIP as the protocol to be extended for deployment in a decentralized fashion. We utilize an existing signaling protocol, properly modifying its semantics where needed, rather than defining a new one, for reasons of compatibility with already existing applications. SIP has now already imposed itself in the research and industrial communities and is expected to become yet more popular, especially if IP telephony definitely takes off.

The reasons why we deem that signaling must be distributed have already been mentioned above. Without a decentralized architecture for signaling, it would not be possible to establish communication among peers, to set up the needed parameters. Pervasive communication could not be deployed. This dissertation does not address issues related to the communication itself, but to the signaling needed for instantiating a direct communication session between peers. We again stress the fact that direct communication does not substitute network-relied communication, but rather it extends it.

We have chosen SIP as the signaling protocol, because it is now well understood both in the academic and in the industrial fields. SIP has been chosen by 3GPP [142] as the signaling protocol for its IP Multimedia Subsystem [1]. According to our idea of pervasiveness of computing facilities among all the end devices, it was necessary to employ an as lightweight signaling framework as possible, so that also the smaller portable devices could easily implement it. The SIP protocol ideally matches this assumption, due to its structure that explicitly allows extensions to a baseline, lightweight protocol for provisions of more advanced services.

Network support in SIP is constituted by SIP servers, maintained by operators or organizations, which SIP end users must use in order to contact remote parties. If SIP server capabilities are enabled in all end devices, it is possible to perform in a decentralized fashion the same operations that are normally executed with the support of centralized servers. The distribution of SIP server capabilities among all end devices is the core element of the proposed architecture for decentralized SIP-based signaling.

A major goal of our approach was to guarantee compatibility with centralized operations. The vision we propose does not exclude the presence of network services, but it rather suggests a seamless interworking between network-provided and ad-hoc services. In other words, decentralized SIP operations must accom-

pany and extend the standard signaling procedures, and not substitute them.

Finally, in order to guarantee interoperability, another major design goal was to yield minimal modifications to the operations of the SIP protocol itself. No new extensions or messages are defined, but the existing methods are rather utilized so that the end devices can transparently exploit decentralized or centralized capabilities.

The philosophy of operations of our approach is to disseminate the user's contact information upon joining a network of peers (like ad-hoc networks). The nodes in the network store the contact information of the newcomer node and reply to it sending their own contact information. Replying and storing is not mandatory, but depends on local policies. For example, a device may have a local security policy to store information of and reply only to previously known users, e.g., friends.

Once the device has the contact information of other users in the network, sessions can be initiated and communication established, in a totally decentralized way. The data exchange only involves the end terminals. The attractiveness of this approach depends on two factors. One is the efficiency of the signaling operations, in terms of low overhead, bandwidth consumption, speed of session establishment, and transparency towards the user. The other is that good and functional applications are built on top of the signaling framework. Secure session management operations, especially in environments where no centralized entities are present to enforce security of exchanged signaling messages, are another key point. Ad-hoc applications should preferably have built-in security mechanisms to cope with the lack of trusted entities.

The main concept of distributed session management has been extended in several directions. Security support has been added to the framework. In particular, our security extensions allow verifying the identity of a sender of a SIP message, and protect the integrity of SIP messages. Confidentiality has been left as an option choice for the applications on top of the SIP signaling middleware. Security support is provided using standard extensions to SIP, modified for use in a decentralized fashion.

The signaling concept sketched above is optimized for use in small ad-hoc networks, where nodes are possibly directly connected with each other at link layer. Using concepts of cooperative data cache and gossiping we have extended the basic concept and enhanced the framework so that the identity of SIP users can be efficiently managed also in bigger ad-hoc networks, with several hundreds of nodes, and multiple hops connecting them.

Finally, the distributed session management framework has been extended in order to achieve interoperability with fixed networks[1]. The Internet, and the na-

[1] In the rest of the dissertation, along with fixed networks, we will use the term *infrastructured*

tive Internet applications are the main example of infrastructured networks decentralized SIP aims to interoperate with. We have designed and experimentally validated a SIP gateway allowing SIP signaling between ad-hoc networks with private addressing space and native SIP applications in the Internet. The design is completed by an application level relay that permits instant messaging sessions to be established in heterogeneous environments. The resulting framework constitutes a flexible and effective approach for the pervasive deployment of real-time applications.

1.4 Related Work

Not much work has yet been done on the decentralized deployment of SIP. The most important effort towards server-less usage of SIP is described in an Internet Draft [13], individual submission to the IETF SIPPING Working Group. The draft proposes to decentralize SIP registration operations by means of distributed hash tables (DHT); the approach removes the need for centralized entities in the SIP architecture. The DHT algorithm is run over an overlay network based on Chord [123] to spread user registration information over multiple nodes. However, such an approach is not targeted to mobile ad-hoc networks, as the maintenance operations and the look-ups related to DHT result in a large amount of SIP messages, which is an undesired situation in mobile environments. Rather, like most of DHT-based algorithms, the work aims at deploying SIP without need of servers over a geographically very extended region, even world scaled; that is, the Internet.

Other work [7, 61] specifically addresses modifications to SIP for deployment in ad-hoc networks. The proposals share the idea that the exchange of messages is tied to the underlying ad-hoc routing protocol. Our approach does not pose this strong requirement.

Particularly, the paper [61] proposes the use of broadcast SIP messages for spreading the contact information of a user in an ad-hoc network. In order to save bandwidth, SIP messages are sent jointly with the Route Request message of the underlying Ad-Hoc on Demand Distance Vector Routing (AODV) [90] protocol. The idea that can be reused here is to bind SIP messages to any underlying routing protocol. Although attractive because it allows to save messages, the approach is too much dependent on the ad-hoc routing protocol used and may therefore have interoperability problems.

The paper [7] tightly ties SIP-based services, such as Instant Messaging and Presence, with an underlying Cluster-based Routing Protocol. The focus on the paper is more on the creation of the routes and the clusters rather than on the

networks, to indicate networks where support from an infrastructure, such as operator-managed servers, is available, as opposed to ad-hoc networks, which are completely server-less networks.

interoperability with SIP services and it cannot be easily applied to our proposed framework. Our approach is in fact more focused on the provision of services, particularly signaling, rather than integration with underlying routing protocols. We assume the presence of an underlying ad-hoc routing protocol for correctly forwarding SIP messages in the ad-hoc network.

Other works instead present instant messaging and presence services in ad-hoc networks [31, 39, 130, 141]. The approach followed differs slightly from ours, as the presented systems are not modifications to standards like the one we propose, but rather constitute proprietary architectures as no interoperability efforts with existing instant messaging and presence services were presented. Although explicitly designing a system for a particular environment like ad-hoc networks may be more efficient than adapting an existing system, we believe that the possible loss of interoperability with mainstream systems is too high a price to pay, to ensure the success of pervasive communication. For this reason we have set interoperability with the standard protocols as a central design goal for our decentralized session management framework.

Jabber (also known as Extensible Messaging and Presence Protocol, XMPP) [109, 110] is an XML-based instant messaging and presence protocol for the Internet. Jabber is designed as a standard client-server protocol, where clients register to their server (in a model very similar to the one used by the e-mail system) to communicate their presence information and receive updated presence information from authorized buddies. Moreover, the server is used to relay instant messages to the remote party, or to host a chat room. This approach is strictly server-centric, as the jabber server plays a central role in the communication; however, extensions are being designed to allow server-less jabber messaging in link-local environments [111] using zero-configuration networking. User discovery is done using multicast DNS [19], after which presence and instant messages exchange follows standard jabber procedures. Jabber has also been extended to provide signaling for audio and video calls over the Internet [71, 72, 112].

Link-local extensions for jabber is the work that most closely resembles the session management framework presented in this dissertation. However, there are a few features missing in link-local jabber, the most important of them being the impossibility of operating in heterogeneous environments and connecting to the Jabber network in the Internet ([111], Section 1). Moreover, it has not been specified how signaling procedures for audio and video calls could succeed in link-local environments. In general, Jabber is not yet an Internet telephony solution as mature as SIP and does not provide the SIP advanced telephony services. In the future, it will be interesting to see how Jabber will evolve also as an Internet telephony signaling protocol.

1.5 Results

Our implementation of decentralized SIP has proven to fulfill the design goals for which it was realized. The main target of enabling SIP-based signaling operations in a decentralized fashion was demonstrated. Users in a mobile ad-hoc network, elected as an example target network environment requiring decentralized operations, were able to initiate a SIP session and engage in the signaled media exchange.

SIP signaling in ad-hoc networks involves providing end devices with the capabilities for retrieving the identities and contact information of on-line users. In ad-hoc networks, users cannot always expect to be able to contact friends in their contact list, but they must often look for the identities of the users available in that moment in the ad-hoc network. SIP methods have been strategically used for providing such a capability, which is not an immediate feature of the baseline protocol. In server-assisted SIP operations, in fact, it is implicitly assumed that users know the SIP user name of the person to contact, similarly to what happens with e-mail (address) or telephone (number) systems.

The operations in small ad-hoc networks have been generalized to bigger multi-hop networks. We have designed an algorithm for efficient data dissemination, replica, and retrieval in ad-hoc networks. The algorithm allows nodes owning a data item to disseminate it in an ad-hoc network so that the item is replicated and distributed in such a way that any node that subsequently requests it can retrieve it ideally from its 1st hop neighbors. The general algorithm has been applied to decentralized SIP, the disseminated items being the SIP contact information of users in the network.

The interoperability of decentralized SIP with the baseline protocol has been shown by building a gateway to an external network. Sessions could be successfully initiated between users inside and outside the ad-hoc network. Users outside the ad-hoc network utilized standard SIP devices, while ad-hoc users employed the extended protocol functionalities for decentralized use. Besides establishing SIP sessions, we have also experimentally shown how instant messaging sessions can be performed between users in an ad-hoc network, with private addressing space, and users in the Internet. Extending the idea, we can say that we have built an alternative solution for NAT traversal for SIP signaling.

As mentioned before, the framework is secured. We allow users who share a security association, such as a self-signed certificate, to prove each other's identity when exchanging messages in ad-hoc networks, where no centralized and trusted third parties can be used to verify the validity of a certificate. With our approach, one can be sure that he is really communicating with the user that is claiming to be the sender of a SIP message. Additionally, we allow users who do not share a certificate to exchange it on the run, which enables securing SIP sessions from

third-party hijacking attempts. Accessing the gateway is also secured. The SIP gateway (and in theory, the instant messaging relay) can provide access to its services only to authorized users, while all communication between ad-hoc nodes and the gateway is encrypted building IPSec tunnels between them.

The dissertation also presents an analytical evaluation of the signaling overhead of decentralized SIP, for small and big ad-hoc networks. It is desirable when wireless devices are involved, to keep the signaling overhead as low as possible. The results show that the overhead of decentralized SIP is not a major limitation and that the dissemination algorithm can be efficiently used to advertise and retrieve SIP contact information in multi-hop ad-hoc networks. Simulations have been carried out to evaluate the performance of the extensions of decentralized SIP to multi-hop ad-hoc networks, and its results illustrated and discussed as well.

1.6 Research History

The work presented in this dissertation has mainly been carried out within a two-year Finnish national project, Seamless Service Interworking in Heterogeneous Mobile Ad-Hoc Networks, SESSI. This project had three academic partners, the University of Helsinki (UH), Helsinki University of Technology (TKK) and Tampere University of Technology (TUT). The aim of this project was, for the first year, to build a framework of services and achieve their seamless interworking in ad-hoc networks. In the second year, the framework was extended to achieve seamless service interworking between ad-hoc and infrastructured networks, such as the Internet or 3G.

Each university developed a service concept, but the three independently built concepts were designed with common interworking in mind. TKK developed security services, TUT service discovery, while UH session management. The research work leading to this dissertation has been developed within the UH subproject, where the author of this dissertation held the main role in designing the distributed session management framework, implementing it and integrating it with the other SESSI framework components.

The extensions of decentralized SIP for multi-hop ad-hoc networks were instead developed outside the SESSI project, as an addition to the main building block of this dissertation. The aim was to generalize the session management framework to any ad-hoc network environment. The work started thanks to a grant, produced by the MiNEMA European research network, with which the author of this dissertation came in contact with Hugo Miranda, from the University of Lisbon. With Hugo Miranda, we designed the PCache algorithm together, while the development work was on his side. The author of this dissertation also mainly designed the way in which the PCache protocol can be used together with

decentralized SIP. The MiNEMA network also funded a visit to the University of Lisbon to finalize the algorithm, fine tuning its parameters.

This dissertation is based on the following publications:

The software architecture for decentralized SIP, as well as its operating principles have first been presented in a conference paper [67]. The author was the main contributor of this paper.

Another conference paper [66] is the first SESSI project common paper. The project architecture is summarized in the paper, especially from the component interactions point of view. The author of this dissertation wrote to the parts relating to decentralized SIP and its interworking with the service discovery framework, besides contributing to the common parts of the paper.

The journal publication [56] presents the project work and extends the previously published conference papers [66, 67]. The paper provides a thorough background research on related work, a new concept for securing session management in ad-hoc networks, and a performance evaluation of the framework. The author acted as editor of the paper, main writer for the session management related sections, and was in charge of defining and writing the evaluation cases and sections. He also contributed to the common sections of the paper and reviewed it.

Two other papers [77, 68] illustrate the work done by the author in the second year of the SESSI project. The first [77] describes how to communicate between ad-hoc networks with private addressing space and the Internet, both for SIP signaling and the signaled instant messaging application. The author was the main designer of the interworking session management framework and wrote the solution section as well as contributing to the other general sections of the paper.

The second article [68] presents extensions to the decentralized session management framework. The extensions refer to advanced security support and implementation of the SIP-based instant messaging application. Advanced security support refers to the possibility of securing SIP signaling also when the two involved parties did not share a security association in advance. The author was the main designer of the framework and main author of the paper.

A second joint SESSI project journal paper has been written by all the relevant stakeholders and is under review at the time of writing this dissertation. The paper illustrates the work done in the second part of the project, discussing how the SESSI service framework can be enabled in heterogeneous environments by means of a gateway. Additionally, the paper shows how the three framework services can interwork seamlessly.

The following papers refer to the work done with Hugo Miranda and also funded by the MiNEMA project. A joint technical report of the University of Helsinki and University of Lisbon [82] introduces the PCache algorithm used for

disseminating and retrieving cached data items in ad-hoc networks. An analytical evaluation and a simulation-based performance study are also provided to complete the publication. A reduced version of the algorithm is also published as part of a book chapter on epidemic dissemination algorithms for data storage [6]. In the conference paper [81] the procedure used in PCache for deciding how to process a received PCache message is presented in more details. The decision is based on several factors, such as number of retransmissions of the same message listened to during the message hold time. In all these publications, the role of the author was of co-designing the algorithm together with Hugo Miranda, and contributing in writing the publications.

Finally,the paper [69] describes how PCache can be applied to dSIP, in what we have called SIPCache. The paper describes how SIP native messages can be mapped onto PCache messages, and proposes an extension to PCache for performing profile-based queries. A performance evaluation of the resulting protocol is also provided. This paper is a milestone in the process of extending dSIP to multi-hop ad-hoc networks, as it describes how PCache can be used to manage SIP users' contact information in ad-hoc networks in a distributed and efficient way. Implementing SIPCache, nodes are able to search for and retrieve SIP addresses in ad-hoc networks without need to know all of the users in the network; the information is made distributed by SIPCache. The author of this dissertation was the main designer of the way in which PCache can be applied to decentralized SIP, while the paper [69] was co-authored with Hugo Miranda.

PCache is a continuously evolving algorithm. The author is currently supporting the main author Hugo Miranda in the work of improving it and refining its parameters. A paper presenting a modified version of the PCache algorithm illustrated in this dissertation has been accepted for publication at a conference [79]. Moreover, another paper is pending review at the time of writing the dissertation. A paper describing how the algorithm can be applied to sensor networks has been accepted to complement the proceedings of OPODIS 2006 [80]. This dissertation presents the original version of PCache, which has been used as a basis for the SIPCache service location mechanism; discussing the modifications to PCache is out of the scope of this dissertation.

This dissertation glues together the work presented in the previously mentioned papers, harmonizing it into the structure of a single monographic discussion. On top of this, the dissertation also extends and deepens some of the topics presented in the published articles, especially when presenting the evaluation of the designed solution.

1.7 Structure of the Dissertation

The remainder of this dissertation is divided into two parts: the first part provides an overview on the technologies related to session management in mobile decentralized environments. The second part presents our solution for decentralized deployment of SIP, and discusses its implementation.

Chapter 2 discusses SIP, the signaling protocol utilized for decentralized ad-hoc deployment in more detail. The extensions that fit our framework best are presented and a practical use case where SIP is used not only as signaling protocol but also as the protocol for user media data exchange is provided.

Chapter 3 deals with the network environments where we envisage to deploy decentralized SIP. Ad-hoc and proximity networks and its related issues as a distributed network environment are extensively addressed. The chapter gives a discussion on other services needed, besides session management, in ad-hoc environments, such as gaining IP connectivity. Finally, the chapter presents future deployment scenarios for WLANs, used as an example wireless technology to form stand-alone ad-hoc networks or for the interworking with infrastructured networks.

Chapter 4 addresses techniques for the dissemination of information in distributed environments. The goal of dissemination techniques is to spread user information to the highest possible number of other users, with a reasonably limited overhead. When a central repository where storing and retrieving information of other users is missing, efficient data dissemination among multiple nodes is of utmost importance.

Chapter 5 examines thoroughly the design of decentralized SIP. Two alternative approaches for retrieving users identities in an ad-hoc network are presented and compared and an algorithm for deploying decentralized SIP in big multi-hop ad-hoc networks is illustrated.

Chapter 6 provides considerations on the prototype implementation of our signaling framework. The chapter also discusses the software modules of which our solution is comprised and presents an example of messages signaling exchange.

Chapter 7 discusses security implications of decentralized SIP and a solution for securing its operations.

Chapter 8 describes the SIP gateway, an entity designed to allow interoperability and communication between devices in ad-hoc networks running decentralized SIP and devices in the Internet running the standard protocol.

Chapter 9 illustrates the applications that we built to demonstrate an example set of services that the decentralized SIP session management framework enables. We built a dedicated proximity manager application that fully utilizes the middleware capabilities. The proximity manager application is used to launch a dedicate instant messaging and presence application and more advanced security services

for ad-hoc networks. Interoperability with standard Internet applications is guaranteed.

Chapter 10 analyzes the implementation results of decentralized SIP, and presents an analytical evaluation of its signaling overhead, both for proximity and multi-hop ad-hoc networks.

Finally, Chapter 11 summarizes the main issues raised by the dissertation, and concludes by presenting pointers for future work.

Part I

Background Overview

Chapter 2

Inside the SIP Protocol and Farther Away

This chapter describes the main characteristics of telephony signaling protocol that we have utilized for decentralized server-less deployment in ad-hoc networks, the Session Initiation Protocol [106]. Before describing the protocol, we explain what it is meant with session management and signaling. This chapter presents the format of some relevant SIP messages and shows how they are used in two example SIP usage scenarios. In addition, the chapter presents the extensions to the baseline protocol that are most interesting for our purposes and example of applications that use SIP.

2.1 Session Management and Signaling

When two or more users want to communicate, various parameters necessary for establishing connectivity must be exchanged among the involved parties. The kind of parameters to be communicated depend on the type of communication, on the characteristics of the end devices used for communication, on the media used in the communication and so on. This exchange of information is generally referred to as *signaling*.

Signaling a phone call in the public switched telephone network (PSTN) involves setting up a dedicated circuit to interconnect called and calling party. If a third user wants to call one of the involved parties, a busy tone is generated as all the resources available for each of the end parties are entirely reserved for the ongoing conversation. Signaling for Internet Telephony involves, among others, communicating to the remote party the IP address and the port where the media stream can be received, the types of media that a receiver can process, and how they are coded. Signaling does not refer only to parameters affecting directly the

communication; for example, signaling can be used in the Internet for reserving resources in the routers so that adequate support for communication, i.e., Quality of Service (QoS), can be guaranteed.

A signaling protocol is needed for carrying the signaling information. In PSTN networks, the Signaling System Number 7 (SS7) protocol suite is used; in the Internet, two telephony signaling protocols have emerged, SIP [106] and H 323 [47]. Signaling in the Internet is not only limited to telephony, i.e., QoS parameters can be negotiated using signaling protocols. An Internet protocol for signaling QoS is the Resource Reservation Protocol (RSVP) [12]. This dissertation specifically addresses issues related to Internet Signaling for setting-up and managing media sessions.

The concept of session broadens the idea of signaling exclusively for Internet telephony services. A communication session among two or more users does not refer only to an Internet telephony session, but to any kind of communication. A session can be established for enabling a multi party conference, involving the exchange of video and audio streams or to exchange instant messages. Session management refers to the procedures necessary to set-up, control and terminate multimedia communication sessions.

Schulzrinne and Rosenberg [115] define Internet Telephony (IP telephony) as *"the real-time delivery of voice (and possibly other multimedia data types) between two or more parties, across networks using the Internet protocols, and the exchange of information required to control this delivery"*. Signaling is an integral part of the Schulzrinne and Rosenberg telephony framework and referred to as information whose exchange is needed to control the delivery of multimedia data. The reader is redirected to the afore-mentioned article, where the differences between Internet telephony and the circuit-switched telephony system are pointed out. The advantages of Internet telephony are pointed out as well.

It is general knowledge that the most important strengths of IP telephony lie on the possibility of deploying more advanced services and communication media than circuit-switched telephony, and on the efficiency of network resources utilization. In IP telephony a circuit is not reserved for the whole duration of the communication, but resources are physically occupied only when a network node must transmit the packet. However, circuit-switched telephony provides high quality of communication (guaranteed by advance resource reservation when setting-up the circuit) and massive deployment.

A most effective way to ensure worldwide diffusion for IP telephony is to provide users with the same quality that "plain" telephony can ensure and guarantee interoperability so that calls can be placed between IP and PSTN networks. While the second task can be reached by adopting gateways accommodating the differences between the two networks types, ensuring QoS in a best-effort network like

the Internet is a big task. Despite the efforts and the proposal matured in the IETF, a final solution is still far away; user mobility adds complexity to this scenario.

IP telephony is at the moment a possibility for the future; it is not yet a technology as widespread as circuit-switched telephony, or cellular telephony, although it is a steadily growing technology, both in maturity and number of users. Neither, if considered as an Internet-based service, it has reached the level of diffusion of services like e-mail or the World Wide Web. However, its diffusion as an Internet service is costantly growing, as shown by the number of providers that offer SIP-based telephony service to PSTN networks. This type of service allows calling from a so-called softphone, i.e., a SIP-enabled software, to PSTN numbers at cheap prices, lower than a pure PSTN call would be. Such a service is enabled by deploying gateways that translate SIP signaling and Internet packet-based media traffic into the format used in the PSTN networks.

There are also several efforts, some of them quite successful, to provide basic telephony services purely in the Internet. One of the most popular VoIP solutions is Skype[1]. Skype is a proprietary technology that utilizes a peer-to-peer approach to allow real time multimedia sessions to traverse firewalls and NATs, transparently for the end user. A key value of Skype is that it allows to place calls within the Internet for free (and to the PSTN network at low fares as well). A technical report from Columbia University [9] analyzes the characteristics and performance of Skype in detail based on studies on sniffed Skype network traffic. Skype communication is encrypted.

A different approach is chosen by GoogleTalk[2] GoogleTalk allows real time multimedia sessions using a totally open protocol, Jabber (XMPP)[3]. In this way, clients other than GoogleTalk that implement the Jabber protocol can connect to the GoogleTalk server and engage in multimedia sessions with other users. GoogleTalk uses extensions to the main Jabber protocol to signal an audio session [71, 72]. Although the GoogleTalk client itself does not support video calling, extensions to the Jabber protocol for establishing video call sessions [112] have been implemented into commercial devices[4], which can be used to place Jabber-based video calls over the Internet. Jabber was born as an instant messaging and presence protocol, so its deployment as an Internet telephony signaling protocol is another signal of the growing interest in communication as an Internet service.

[1] Skype: Free Internet telephony that just works. At `http://www.skype.com/`, accessed October 9, 2007.

[2] GoogleTalk: Connect with your friends for free. At `http://talk.google.com`, accessed October 9, 2007.

[3] The Jabber Software Foundation: `http://www.jabber.org`, accessed October 9, 2007.

[4] Nokia N800: Take the Internet into new places. At `http://www.nokia.com/n800`, accessed October 9, 2007.

2.2 Session Initiation Protocol

The Session Initiation protocol (SIP) [106] is the Internet Engineering Task Force (IETF) [143] proposal for internet telephony signaling. RFC 3261, released in June 2002, describes the current SIP specification, and obsoletes the first specification of the protocol, RFC 2543 [74], dated March 1999. This first version was itself the result of a three years long standardization period, begun in the Multiparty Multimedia Session Control (MMUSIC) IETF working group. The revision and development process of SIP led to thirteen Internet Drafts (ID), before the protocol was approved as RFC 2543.

After that, the protocol has gained so much interest in the academic and industrial world, and in the IETF itself, that a dedicated IETF working group (SIP) has been chartered to standardize SIP related topics. Besides the SIP working group, several other WGs are addressing issues directly or indirectly related to SIP; many RFC and IDs have been produced so far and are still being produced. The success among developers has not yet translated so far into a strong commercial deployment. Although the implementations of SIP, both open source and commercial, are numerous, and the protocol has been chosen by the Third Generation Partnership Project (3GPP) [142] as the signaling protocol for its IP Multimedia Subsystem (IMS) [1], SIP is still too young to be universally deployed like the TCP, UDP or IP protocols. In the Internet, there still are several proprietary signaling solutions, or proprietary modifications to the baseline SIP standard; the path to a fully widespread utilization is still long.

2.2.1 Introduction to SIP

SIP is *"an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants"* [106]. Examples of sessions that SIP can control are Internet telephone calls, and multimedia conferences. However, the signaling framework utilized by SIP is general and can be used for any kind of session in the Internet. The IETF SIMPLE working group[5] is defining extensions to the SIP protocol for enabling Instant Messaging and Presence Leveraging Applications. In the SIMPLE framework, SIP is used for signaling instant messaging sessions, which are mainly composed of text messages, but can be enriched with multimedia flows.

SIP allows users to communicate to the remote party the parameters describing the session being signaled; the remote party, based on the session description received, decides whether to accept, refuse or accept with modifications the session initiation request, returning another session description. SIP is not bound

[5]IETF SIMPLE working group: at `http://www.ietf.org/html.charters/simple-charter.html` accessed October 9, 2007

to any method for describing the parameters characterizing a session; however, the Session Description Protocol (SDP) [43], an independent protocol developed by IETF for describing multimedia sessions, is generally used for carrying SIP session descriptions.

SIP bases its signaling framework on the offer/answer model [103]. According to this model, a SIP user (the offerer) can invite a remote party to a SIP session offering an SDP description. The remote party, the answerer, returns a session description containing its own parameters. A variation to the basic offer/answer scheme is that a SIP invitation does not carry any offer; in this case the offer will be presented by the invited party, if it accepts to begin a session, and the calling party will answer to the offer.

SIP can run on top of several different transport protocols, either reliable or unreliable. SIP is a client/server protocol, text-based, which reuses much of the syntax and the philosophy of HTTP [34]. The fact that SIP is client/server means that SIP operations are always in the form of a SIP entity (the client) that generates a request, and a receiving entity (the server), which processes the request and generates a response. As in HTTP, SIP requests are done invoking specific SIP *methods*.

The distinction between client and server in SIP is logical; a same entity can act as client in a SIP session and at the same time as server in another session. SIP entities generating request and responses are referred to as *user agents*. There are however SIP entities acting as servers in a more classic sense, providing services to clients. The role of SIP user agents and servers in the SIP architecture is described in the next subsection.

An overview on the basics of the SIP protocol is now provided. A most authoritative SIP overview paper, although describing the first version of the protocol, is provided by two of the SIP main designers, Schulzrinne and Rosenberg [116].

2.2.2 SIP Functional Entities

The Session Initiation Protocol has four basic logic functional entities. Extensions to the baseline SIP protocol may define additional entities:

- Registrar Server
- Proxy Server
- Redirect Server
- User Agent

Registrar, proxy and redirect server logically differ in the operations they perform; however, it is not uncommon that a network operator may decide to colocate

these three entities into a single machine. Usually, SIP servers are maintained by operators or network providers, handle a specific domain, and take care of performing SIP related operations for the users registered in their domain.

Registrar Server

A registrar server is the SIP logical entity that accepts SIP registration requests and maintains a database where the contact information of all the registered users is stored.

Proxy Server

A proxy server is an intermediary entity that can act both as a server or a client for the purpose of making or forwarding SIP requests on behalf of SIP endpoints towards the final destination on the SIP network. It is mainly an application layer router.

Redirect Server

A redirect server generates responses to requests it receives, directing the calling client to another location where the called client can be reached.

User Agent

User agents are usually the endpoints of the SIP network. They generate requests, or responses to the requests they receive. In the first case, they are referred to as User Agent Clients (UAC), in the second, User Agent Servers (UAS). A SIP server can act as a user agent; for example, when a redirect server sends a response to a UAC indicating the new location where the called user can be contacted, the server logically acts as a UAS. A user agent conceptually refers to the devices utilized for exchanging SIP messages; the same human user, can in fact register into the SIP network from two different user agent devices, i.e. desktop computer and PDA.

2.2.3 SIP Services

Unlike H 323, SIP was not primarily designed with the main objective of integration with the older PSTN system and provision of a telephony signaling framework able to support a service as similar as possible to circuit-switched telephony. SIP rather aims at defining a general service framework on top of which various kind of multimedia applications can be built. Nevertheless, one of the main target applications was Voice over IP.

The most common way for designing and implementing additional services in SIP is defining *extensions* to the baseline protocol, in the form of new methods, header fields or parameters. A SIP request, which is an extension method itself, or is carrying an extension, is examined by servers or UASs; if the extension is not implemented, the request is processed ignoring the extension or it is discarded and the requestor is asked to submit the request without the unsupported extension. This approach allows building lightweight UAs, implementing only the SIP features needed by the application.

Most of the supplementary services provided by H 323 are supported by SIP as well. While in the H 323 platform all the supplementary services are described in the series H 450 of recommendations, in SIP they can be exploited by the baseline protocol itself, or are described as extensions in separate documents. For example, support for call transfer is provided in SIP with the definition of a new method, REFER, defined in RFC 3515 [120]. Call forwarding instead, can be implemented in SIP using final responses of type 3xx, indicating that the call is redirected to another location. Most of the advanced call control features for SIP are being defined in several IDs and RFCs by the IETF SIPPING working group. We direct the interested reader to the web page of the charter for further details [6].

A SIPPING ID provides examples of SIP services, summarizing the work ongoing for enhancing SIP with advanced call control services [55]. Table 2.1 lists few examples of SIP advanced telephony services. In the extension column it is reported the SIP extension enabling the service; it is possible to see that the baseline protocol is able to provide some advanced telephony services, like Call Hold or Call Forwarding. The REFER extension is used to exploit Call Transfer and Call Park, which in SIP is treated as a particular case of Call Transfer. RFC 3515 [120] describes the REFER extension from a general point of view; separate documents describe how to use the extension to provide the actual service. For example, the usage of REFER to provide Call Transfer services is described in a SIPPING WG Internet Draft [121].

Table 2.1: Examples of SIP advanced telephony services

Service Name	**Extension**	**Specification**
Call Hold	Standard SIP	RFC 3261 [106]
Call Transfer	REFER	RFC 3515 [120]
Call Forwarding	Standard SIP	RFC 3261 [106]
Call Park	REFER	RFC 3515 [120]
Call Pickup	SUBSCRIBE/NOTIFY	RFC 3265 [99]

[6] IETF SIPPING Working Group `http://www.ietf.org/html.charters/sipping-charter.html`, October 9, 2007

SIP has an inherent support for personal mobility; it was designed with a wider scope than H 323, for wide area networks rather then for LANs. H 323 was only afterwards adapted for use in WANs with the definition of zones. Support for personal mobility in SIP is given in several ways: the protocol allows for call forking, that is, forwarding the same call to multiple destinations. Call forking is provided by baseline SIP; proxy servers effect forking by sending simultaneously the same SIP request to multiple user agents belonging to the same user. SIP also provides support for mid-call mobility, by defining mechanisms for coping with change of the parameters negotiated at the beginning of a session. Such parameters could be a different IP address where a user agent can receive media, or added support for a new media flow. The re-INVITE mechanism, described in Section 12.2 of RFC 3261 [106] is used for the purpose.

2.2.4 Security Mechanisms for SIP

Security mechanisms for SIP are defined by the main specification and by later extensions. The general idea is to reuse existing security solutions and adapt them for SIP. A good introduction to the logic of SIP security mechanisms can be found in a paper by Salsano et al. [113]. In short, security mechanisms for SIP can be classified into hop-by-hop and end-to-end.

Hop-by-hop mechanisms secure communication between two successive SIP entities in the SIP signaling path. For example, they are applied to the connection between a user agent and its serving SIP proxy server. Hop-by-hop mechanisms are not enough for SIP as there is no assurance that in the next hop of the SIP routing chain the security mechanism used so far will be enforced, or even that security will be used at all.

End-to-end mechanisms involve directly the endpoints of a SIP call, usually the UAs. An example of end-to-end security mechanism is encryption of messages. SIP is a peculiar protocol from the point of view of security, as it comprises intermediary entities, which can modify a SIP message on its route to destination. For such a reason, hop-by-hop security is needed besides end-to-end. For example, it is not possibly to fully encrypt SIP messages, as proxy servers need to examine some header fields, like Via or Record-Route, for correct application layer routing. Security in SIP involves mainly authenticating users accessing a SIP service, like UAs do with proxies or registrars, and ensuring integrity or confidentiality of exchanged data. Extensions to the baseline protocol define further levels of security, like privacy [92]. We now describe some of the security mechanisms envisaged for SIP.

IPSec

IP security (IPSec) [59] provides hop-by-hop security. Since SIP is an application layer protocol, the usage of IPSec can be done transparently for the SIP applications; its usage with SIP is therefore recommended in those scenarios where application level security is complicated or non worthy. Another scenario where IPSec could be used is when UAs have a pre-shared keying relationship with their first-hop proxy server. The usage of IPSec with SIP is introduced in the main specification [106], Section 26.2.1.

Transport Layer Security

Transport Layer Security (TLS) [25] is a transport layer, hop-by-hop security mechanism. Necessary condition for TLS to be utilized is that SIP over TCP is supported by the SIP entities using TLS protection. Unlike IPSec, in fact TLS usage is tightly coupled with SIP. For all the other aspects, the usage of IPSec and TLS are logically similar, both being external lower layer security mechanisms. TLS is most suited to scenarios in which hop-by-hop security is required between hosts with no pre-existing trust association.

S/MIME

S/MIME [96] is a versatile security scheme that can provide end-to-end encryption of SIP messages, integrity and authentication. SIP message can transparently carry any kind of body, and, thus, also S/MIME bodies. SIP messages can carry an encrypted S/MIME body to protect the session description, or they can carry the encapsulated whole SIP message, which provides confidentiality of the body and integrity of the SIP signaling message. S/MIME can be used for authentication or integrity purposes only, if it carries the signature and/or the certificate of the sender. A major drawback of S/MIME is the high size of SIP messages when carrying S/MIME bodies, which can cause fragmentation of SIP messages. Fragmentation is a problem when SIP is carried over UDP, because the non reliable transport protocol does not provide means to rebuild messages that get fragmented at lower layers. The usage of S/MIME with SIP is extensively detailed in Section 23 of the SIP specification [106].

Digest Authentication

SIP provides a stateless, challenge-based mechanism for authentication that is based on the authentication scheme of HTTP [36]. The digest authentication scheme, detailed in Section 22 of the SIP specification [106], provides authentication and replay protection for SIP messages. When a UA sends a request without

credentials to a SIP server, which requires authentication, the server will challenge the UA with a proper SIP response containing the challenge data. The UA will then resend the request answering the challenge and containing the requested credential information. Both user-to-user and proxy-to-user authentications are possible.

2.3 Overview of SIP Operations

This section provides a high level synthesis of SIP operations. A more detailed description, illustrating the messages exchanged and their meaning, is provided later in Section 2.5. In order to exploit SIP services, users must first register their presence in the SIP network. Registration involves communicating to the registrar handling the domain where the SIP user is registered, the possible locations where a user is willing to be contacted. SIP allows specifying a preference order when multiple contact locations are provided. In a standard scenario, SIP messages travel along a chain of proxy servers, until they reach the intended recipient. The reason for this is that SIP users often do not know the location where the called party is reachable, and therefore they must leave the task of locating called users to proxy servers. Nevertheless, direct communication between peers is possible if the calling party knows the callee's IP address.

SIP users are identified by e-mail like identifiers, preceded by a SIP URI, such as:

sip:alice@atlanta.example.com

A SIP user name is referred to as *Address of Records* (AOR) for that user. When a user registers to a SIP registrar, an association between the AOR and the list of provided contact addresses is created in the server database; normally the contact address is an IP address, but it can be anything identifying a network access point, such as a telephone number. The association between a SIP AOR and the list of the contact addresses is called *binding*. The database where the bindings are stored is also called *location service*. Registrar servers are the SIP entities that maintain the location service, while proxy servers exploit its functionalities.

The recipient of a SIP request is indicated in the target-URI header field, present in every request. When a proxy receives a SIP request, it processes the AOR indicated in the target-URI and strips out the domain part of the AOR. If the proxy is not in charge for that domain, then it resolves the address of the proxy (methods for doing this are specified in RFC 3263 [104]) responsible for that domain, and forwards the request to the resolved address. If the proxy is responsible for the domain specified in the target-URI, it consults the location service to find

out the address where the recipient user can be contacted, and eventually delivers the SIP message to its destination.

SIP is a particular protocol from the application layer routing point of view, as its messages are routed through several intermediate hops before reaching the final recipient. When sending a message, SIP entities need to determine the next application level hop where to forward the message. The next hop in the SIP chain can be different from the final recipient of the message, as the example in Section 2.5.2 will show.

SIP entities often need to use the Domain Name System (DNS) to resolve the name of the next hop entity. For example, a UA may need to resolve the name of its predefined outbound proxy server; in turn, the outbound proxy server may need to resolve the name of the proxy server handling the domain of the called user. RFC 3263 [104] defines the procedures that SIP entities must use for SIP-related name resolution operations. In order to correctly forward a SIP message to the next-hop entity (referred as server in the DNS resolution operations context), its IP address, port, and transport protocol must be determined by the sending SIP entity (referred here as the client). The choice of the correct transport protocol for SIP is particularly important, as the client should always use the server preferred transport protocol to contact the server.

Moreover, since SIP often signals interactive multimedia applications, it is important that the signaling is time efficient. Since DNS searches are hop-by-hop, the time needed for all the DNS searches in the end-to-end path can take a significant portion of the overall signaling time. Considerations on load sharing, and on how to increase the efficiency of DNS look-ups are provided in the RFC 3263 [104].

2.4 SIP Messages

SIP reuses much of HTTP logic, its syntax and the way messages are built and coded. SIP messages are distinguished into *requests* and *responses*; SIP is a transaction based protocol, and a transaction is independently formed by a request and its related responses.

SIP requests are carried out invoking a *method*; similarly to HTTP, a request message carries in its first line the method it invokes, an URI indicating the target of the request and the SIP version in use. The syntax of responses is also borrowed from HTTP; a SIP response is characterized by a numeric three digit code and a short reason phrase clarifying the meaning of the response. Several of the HTTP status code/reason phrase combinations are reused to facilitate the deployment and understanding of SIP.

SIP responses can be of two types: provisional and final. Provisional re-

sponses indicate to the requestor that the request is being processed by the entity that has generated the response and that it was not lost. This implies that there is no need of retransmitting the request. Final responses, sent by the entity acting as a UAS in the SIP session, communicate the outcome of the request.

A SIP message, request or response, is formed by several header fields and one or more optional bodies. The header fields are arranged in a `key: value` format and may carry one or more optional parameters, very similarly to the HTTP message structure. The SIP specification [106] defines six methods, REGISTER for registering contact information, INVITE, ACK, and CANCEL for setting up and managing SIP sessions, BYE for terminating sessions, and OPTIONS for querying servers about their capabilities. Next sections briefly describe these methods. Extensions to the baseline specification may define additional methods, or header field parameters, and their usage within the SIP architecture.

2.4.1 SIP requests

REGISTER

The SIP REGISTER message, that is, the SIP request message invoking the REGISTER method, is used by UAs for communicating user's bindings to a registrar. Registrars can populate the location service database with the information received from (authorized) users with a REGISTER message. However, the SIP specification does not prohibit to populate the database with other external means. Examples of usage of the REGISTER message will be given later in Section 2.5.1.

A user can register to a registrar from several user agents; sometimes, user agents perform registrations or unregistrations automatically, so that it may be difficult to keep track of all user's active registrations. The SIP specification allows to use REGISTER to receive the list of all the current contact addresses registered for a given AOR. This is done by sending a REGISTER message without Contact header field.

INVITE

The SIP INVITE message is used to begin or modify SIP sessions. An example of session initiation using INVITE will be given in Section 2.5.2. INVITE is a particular kind of request, and its handling differs from all the other SIP requests. The reason is that INVITE requests can be replied after a relatively long amount of time, depending on when the called user physically answers to the call. For example, a user could momentarily be away while his UA displays an incoming session invitation; a final response to the INVITE message could therefore arrive several minutes after having issued the request. For all the other requests SIP expects a final response within a relatively short time frame.

A response to particular types of SIP requests, such as INVITE, forms a *dialog*. A dialog is a peer-to-peer SIP relationship between two UAs that persists for some time. A dialog is identified by 3 elements: a Call-Id, a local tag, and a remote tag. Call-Id is a SIP header field, while the tags are parameters of the From and To header fields. Once all the necessary elements for referring to a dialog have been stored, a UA can send requests inside the context of that dialog; all these requests must have the same Call-Id, and local and remote tag as those that have formed the dialog.

An INVITE can be sent within a dialog by either party for example to modify session parameters; e.g., a user may notice that he is running out of batteries and drop the video stream from the conversation. This is done by sending an INVITE referencing to the dialog already established with the remote party containing a SDP description where the video flow is missing.

ACK

An ACK request is sent, always by the calling UA, to complete session initiation procedures, after a final response has been received by the inviting party. ACK requests are sent only as consequences of INVITE requests. The reason is the same as explained above, that an INVITE can take a long time before being answered, as opposed to other requests. If the final response to the INVITE does not indicate success (i.e. it is not a 2xx response), the ACK is considered part of the transaction, otherwise the ACK is not part of the transaction. This is because a 2xx response to an INVITE establishes a SIP session.

ACK requests refer to an existing INVITE transaction by inserting the branch parameter in the SIP Via header field with the same value than the original INVITE. The branch parameter in fact identifies a transaction. ACK requests are considered as requests sent within a dialog, and therefore have the matching tag values in the To and From header fields. Cseq and Call-Id header field in an ACK request match those of the INVITE that has triggered the ACK.

ACK messages can carry a body when they answer to a session offer sent within a 2xx response. This scenario of session invitation can occur when the inviting user does not include a session description in the body of the initial INVITE message.

CANCEL

A CANCEL request is used to terminate a pending request, that is one for which no final response has been received. CANCEL requests constitute an own transaction, but reference the transaction to be canceled: a CANCEL request has in fact the same value of the branch parameter as the request it cancels.

A CANCEL request asks the UAS to cease processing the request and to generate an error response to that request, but it has no effect for requests to which a UAS has already sent a final response. For such a reason, it is practical to issue CANCEL requests only for those requests for which it may take a long time at the UAS before a final response can be generated. In other words, it is practical to cancel only INVITE requests. Section 9.1 of [106] describes how and in which condition it is possible to generate CANCEL requests.

BYE

A BYE request is used to terminate an established session within a dialog. The usage difference between BYE and CANCEL depends on the status of the session establishment process. The caller UA can issue a CANCEL request if it wants to terminate a session that has not been fully established yet, and for which a final response has not been received yet. Otherwise, a BYE request should be issued.

The called UA cannot issue CANCEL requests, but only BYE requests. However, since BYE requests can cancel only established sessions, they can be issued only after the ACK confirming the dialog establishment has been received from the calling UA. A BYE request must be replied with a proper response, usually 2xx.

OPTIONS

OPTIONS is a SIP method that can be used to query a server or UA about its capabilities; its usage is detailed in Section 11 of the SIP specification [106]. The OPTIONS method allows knowing whether, e.g., a SIP extension or a media codec is supported by a remote party. Capabilities information can also be obtained using INVITE and SDP session description. However, this would result in the called user to be alerted of a session invitation request, while the calling user was instead only interested to know the remote UA capabilities. The capabilities query exchange with OPTIONS is typically performed only by the user agents; the human users are thus left unaware of the fact that their user agent is being contacted. For example, before a client inserts a Require header field into an INVITE listing an extension that it is not certain the destination UAS supports, the client can query the destination UAS with an OPTIONS to see if this extension is returned in a Supported header field.

The response code to an OPTIONS request is the same that would have been chosen had the request been an INVITE. This allows the calling UA to know in advance whether the called user is ready to accept a call ([106], Section 11). The response to an OPTIONS request can contain a session description of the media types accepted by the queried UA.

2.4.2 SIP responses

SIP entities must always send responses to issued requests indicating that the request is being processed or to report the outcome of the request. SIP provisional responses indicate that a request is being processed and are indicated by the status code 1xx.

When a proxy receives the INVITE, it may send a 100 Trying response indicating to the calling UA that the request has been forwarded to the next SIP hop. Although this information is usually of little use for the human user, which may not be notified of the arrival of a 100 message, the 100 Trying message is useful to tell that there is no need for retransmitting the request. Another provisional response is 180 Ringing, which can be sent by the called UA to tell that the session initiation request has been delivered to the user (the phone is ringing).

Final responses indicate the outcome of a request. The following classes of final responses are defined:

- 2xx: they indicate the success of a request. Example: 200 OK for accepting a request. Additional information is usually carried in the header fields and the body of the response
- 3xx: they indicate redirection of a request. Example: 302 Moved Temporarily. The inviting user should retry to contact the remote party at the address provided in the message.
- 4xx: they indicate that a request has failed, providing information on the causes of the failure. Example: 401 Unauthorized. It is used to request authentication for an issued request and usually provides an authentication challenge.
- 5xx: they indicate internal server failures. Example: 501 Not Implemented. The server does not implement a feature needed to successfully fulfill the request.
- 6xx: they indicate global request failures, meaning that a server has definitive information about a particular user, not just the particular instance indicated in the Request-URI. Example: 600 Busy Everywhere. The server has information that the contacted user is busy in any of the contact addresses registered at the moment and cannot be contacted. This is in opposition to a response like 486 Busy Here, which indicates that a user is busy at the particular instance being contacted, but may be free at another one.

For more details on the meaning of status codes and their usage, refer to Section 21 of the SIP specification [106].

2.5 Basic SIP Call Flows

This section provides simplified call flows for the two most common SIP scenarios, registration and session establishment. The examples are based on those provided in RFC 3665 [53]. In the examples, the actors are SIP users Alice and Bob, their user agents and the SIP servers of the domains where they are registered. Alice's and Bob's UAs and server URIs and IP addresses are shown in Table 2.2.

Table 2.2: Relations among example SIP entities

User	Element	AOR/URI	IP Address
Alice	User Agent	alice@atlanta.example.com	192.0.2.101
Bob	User Agent	bob@biloxi.example.com	192.0.2.201
Alice	Proxy/Registrar Server	ss1.atlanta.example.com	192.0.2.111
Bob	Proxy/Registrar Server	ss2.biloxi.example.com	192.0.2.222

2.5.1 SIP Registration

Registration in SIP is performed by sending a SIP REGISTER message to the registrar server handling the domain where the user is registered. Usually, the address of the registrar is well known or preconfigured in the user agent, or it can be retrieved with DHCP. In this example, Bob can resolve the name ss2.biloxi.example.com to find out the address where to send the registration. The messages exchanged in a simplified registration scenario are shown in Fig. 2.1.

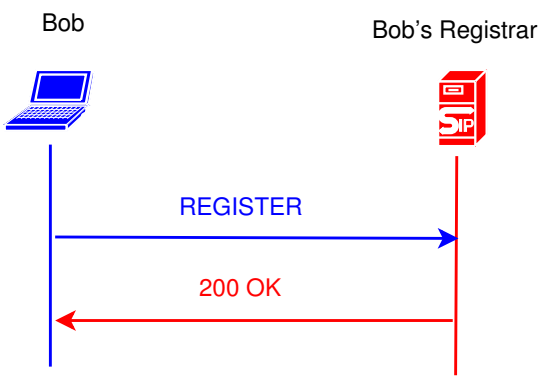


Figure 2.1: Simplified SIP Registration Scenario in the Internet

The REGISTER message Bob sends to his server is the following:

```
REGISTER sip:ss2.biloxi.example.com SIP/2.0
Via:  SIP/2.0/UDP client.biloxi.example.com:5061;
```

```
branch=z9hG4bKnashds7
Max-Forwards: 0
From: Bob <sip:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

The first line of the message contains the request target-URI. For REGISTER request it is the URI of the server where the registration is directed. The Via header field (line broken for visual clarity; the same applies to all the other Via fields showed in Section 2.5) contains the address where the response to that request can be received, besides the transport protocol to be used. In this case, Bob indicates that responses to his registration request must be directed to the machine that has the resolvable DNS name of client.biloxi.example.com, to the UDP port 5061. Alternatively, Bob's user agent could have written its current IP address instead of the resolvable machine name. Branch is a parameter used to identify a SIP transaction.

Max-Forwards indicates the number of application level hops that can process the message; it is the SIP equivalent to the IP TimeToLive field. The fact that it is 0 indicates that Bob's server cannot further forward the request. The From header field indicates the AOR of the user performing the request, while the To header field indicates the AOR of the user whose contact information is being registered with the SIP message. Usually To and From coincide, but they can differ for allowing third-party registrations.

Call-ID and Cseq header fields are identifiers for the transaction the SIP request has initiated. The Contact header field contains the IP address where Bob can receive requests from other entities. In this case, it is in the form of a resolvable machine name, and there are no port numbers specified. Content-Length indicates the length of the body carried by the SIP message when present.

In the example, the server acknowledges Bob's registration with a SIP 200 OK message reporting the success of the registration. If user authentication is needed, the server may challenge the client to provide credentials, sending a different response, usually 401 Unauthorized. Authentication operations depend on the authentication scheme chosen. The 200 OK message has the format:

```
   SIP/2.0 200 OK
Via: SIP/2.0/UDP client.biloxi.example.com:5061;
branch=z9hG4bKnashd92;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=ja743ks76zlflH
To: Bob <sip:bob@biloxi.example.com>;tag=37GkEhwl6
```

```
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq:  1 REGISTER
Contact:  <sip:bob@client.biloxi.example.com>;expires=3600
Content-Length:  0
```

SIP syntax, as already underlined, is similar to HTTP's. Responses contain a three digit status code and a short textual reason phrase. When a request is accepted, SIP entities return the 200 OK code-phrase pair. The Via header field is copied as it was by the registrar, which also adds a parameter indicating the IP address from where the request was received.

Also all the other header fields are copied, to indicate all the contact addresses registered at the server for the user specified in the To header field. If Bob had previously registered to the server from another user agent, the 200 OK message would contain in the Contact field the addresses of both the user agents where Bob can be reached. The Contact header field contains the expires parameter indicating the duration in seconds of Bob's registration at the server.

In order to avoid expiration of contact information at the server, Bob should refresh periodically his binding, by resending a REGISTER message containing the binding to be refreshed. Registrations can be explicitly deleted by a user agent with a REGISTER message carrying the Expires header field set to the value of 0.

2.5.2 SIP Session Establishment

A basic SIP session establishment scenario creates a "trapezoid", as it involves four entities: the two user agents and their serving proxy servers. The proxy server handling the SIP requests generated by a UAC is often referred to as outbound proxy server; usually, its IP address or resolvable name is preconfigured at the user agent. The proxy server forwarding SIP requests to a recipient UAS is referred to as inbound proxy server.

The message chart for a simplified session establishment scenario is shown in Fig 2.2. Alice wants to invite Bob to a SIP session; this is done in SIP using the INVITE method. Since Alice does not know Bob's IP address, she sends the INVITE to her proxy server. Alice's proxy in the domain atlanta.example.com realizes that Bob belongs to a domain it does not handle and forwards the message to Bob's proxy in the domain biloxi.example.com. Bob's proxy delivers the message to destination, as Bob is registered to the domain it handles. Bob accepts Alice's invitation; his response is routed back to Alice through the inverse chain of SIP proxies. When Alice receives Bob's response, her UA acknowledges it sending an ACK message. The ACK message can be sent directly to Bob's UA as Alice knows Bob's contact information from his 200 OK message; this direct exchange completes the trapezoid, as evidenced by the dashed lines in Fig. 2.2. After the three-way handshake establishing the session, the media exchange can

begin. The session is terminated with a SIP BYE message, which is answered with a 200 OK message. Note that an ACK message is sent only when the initial request is an INVITE.

Some of the messages exchanged are shown in the following. The INVITE sent by Alice, M1, is:

```
   INVITE sip:bob@biloxi.example.com SIP/2.0
Via:  SIP/2.0/TCP client.atlanta.example.com:5060;
branch=z9hG4bK74bf9
Max-Forwards:  70
Route:  <sip:ss1.atlanta.example.com;lr>
From:  Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To:  Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq:  1 INVITE
Contact:  <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type:  application/sdp
Content-Length:  151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Those header fields whose meaning is similar to the correspondent present in a REGISTER message are not discussed. In an INVITE request, the AOR of the called party is reported in the target-URI and in the To header field, while the From header contains the originator of the request. These header fields indicate the final recipient of the message; however, the message is actually sent to Alice's proxy, ss1.atlanta.example.com, at the address 192.0.2.111. This is also specified in the SIP message, in the Route header field, which contains the name of Alice's outbound proxy. The INVITE message contains an offer, carried in a SDP body; type and length of the body carried in a SIP message are reported in the Content-Type and Content-Length header fields. The SDP body carries the session descriptor parameters; in this case, the signaled media is audio, which will be carried with RTP. Details on the usage of SDP for describing SIP sessions can be found in [43, 103].

When Alice's proxy receives the message M1, it forwards it (M2). The most notable difference between message M1 and message M2 is in the following

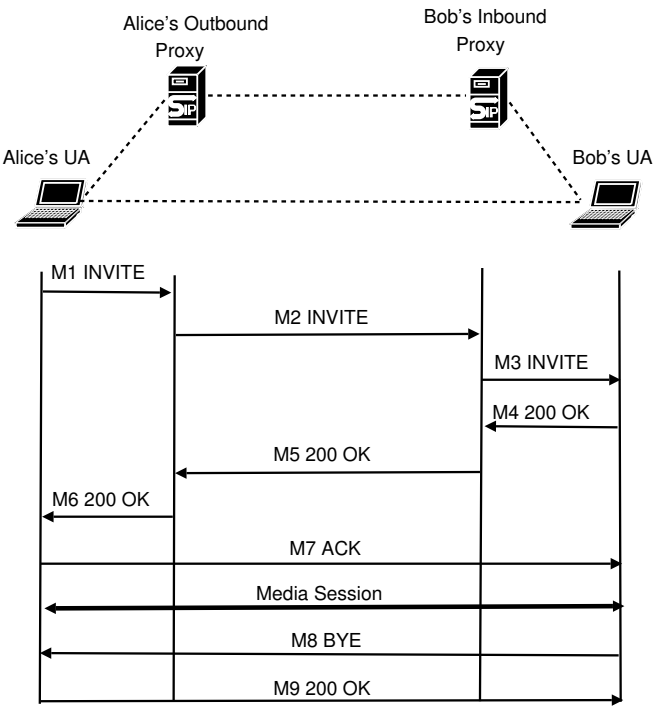


Figure 2.2: Simplified SIP Session Establishment Scenario in the Internet

header fields:

```
   Via:  SIP/2.0/TCP ss1.atlanta.example.com:5060;
branch=z9hG4bK2d4790.1
Via:  SIP/2.0/TCP client.atlanta.example.com:5060;
branch=z9hG4bK74bf9;received=192.0.2.101
Max-Forwards:  69
```

Whenever a SIP proxy forwards a message, it adds a Via header field on top of the others. The response to that request will follow the application layer route defined by the Via header fields. The innermost Via field is the original as generated by Alice's UA, with the addition of the received parameter indicating the IP address from where the server received the request. The second Via is added by Alice's proxy server, to ensure to be in the return path of the response. Max-Forwards is diminished by one. Similarly, the INVITE M3 delivered by Bob's proxy to his UA, will have the following Via header fields:

```
   Via:  SIP/2.0/TCP ss2.biloxi.example.com:5060;
branch=z9hG4bK721e4.1
Via:  SIP/2.0/TCP ss1.atlanta.example.com:5060;
```

```
branch=z9hG4bK2d4790.1 ;received=192.0.2.111
Via:  SIP/2.0/TCP client.atlanta.example.com:5060;
branch=z9hG4bK74bf9 ;received=192.0.2.101
Max-Forwards:  68
```

There is a third Via header field, as added by Bob's server; in the second, the originating IP address is reported, while the first Via is left unchanged. Bob's reply M4 contains an SDP body that answer the received offer:

```
   SIP/2.0 200 OK
Via:  SIP/2.0/TCP ss2.biloxi.example.com:5060;
branch=z9hG4bK721e4.1 ;received=192.0.2.222
Via:  SIP/2.0/TCP ss1.atlanta.example.com:5060;
branch=z9hG4bK2d4790.1 ;received=192.0.2.111
Via:  SIP/2.0/TCP client.atlanta.example.com:5060;
branch=z9hG4bK74bf9 ;received=192.0.2.101
From:  Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To:  Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 3848276298220188511@atlanta.example.com
CSeq:  1 INVITE
Contact:  <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Type:  application/sdp
Content-Length:  147
v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Bob's UA recopies the Via header fields present in the received INVITE (eventually adding in the outmost Via field the IP address from where the INVITE was received). Call-ID and CSeq match those of the received request, as well as the From and To fields. The Contact header fields reports Bob's contact information; the transport parameter in the header field indicates that Bob prefers TCP to be used as transport protocol for SIP for future requests Alice may want to address to Bob. The Via and Contact header fields are not redundant. The first indicates the address where to return the response to an *already issued* request; the second indicates the address where *following* requests can be sent. Such addresses can in principle be different, but in most of the cases they coincide.

This response carries the parameters necessary for Alice to identify the dialog with Bob. The Call-Id is set by Alice's UA, while the local tag is set as parameter

of the From header field. Alice's UA takes the tag value that Bob's UA has added to the To field as remote tag and can therefore univocally refer to the dialog with Bob. Similarly, on the other side, Bob's UA can unambiguously refer to the dialog with Alice; the only difference is that the values of local and remote tag are switched.

When Bob's server receives the 200 OK, it looks the outmost Via header field to see whether it is supposed to process that response; if so, it strips it out and forward the 200 OK to the address indicated in the second Via header field (M5). Similarly, Alice's proxy will strip out the second Via field and forward the message M6 to Alice's UA. The only two Via header fields present in the message M5 are:

```
   Via:  SIP/2.0/TCP ss1.atlanta.example.com:5060;
branch=z9hG4bK2d4790.1;received=192.0.2.111
Via:  SIP/2.0/TCP client.atlanta.example.com:5060;
branch=z9hG4bK74bf9;received=192.0.2.101
```

The scenario is simplified as possible authentication challenges that proxies may send to the UAs are not shown; the SIP provisional responses, indicating that a call is being processed by the involved entities are not shown as well. A complete session establishment scenario is described in RFC 3665 [53], Section 3.2. In that example, the two proxies wish to stay in the signaling path also after that the 200 response is returned to Alice; this means that all the SIP signaling messages sent after message M6 in Fig. 2.2 would be processed by proxy servers. Proxies express such a request by adding their address or resolvable name to the Record-Route header field of a SIP message.

The usage of Record-Route is similar to Via; the difference is that with Record-Route proxies explicitly request to stay in the signaling path for all the SIP requests sent within an established dialog. In the example, if Record-Route would be added, the proxies would have received also the ACK and BYE requests sent after successful session establishment. Staying in the path for a response is instead automatic and mandated by the SIP protocol: all responses follow the reverse path of the requests. The pointer to the next hop where a response should be forwarded is given by a proper Via header field.

2.6 SIP Extensions

Several extensions to the baseline SIP protocol have been proposed so far; this section discusses those most interesting for our decentralized session management architecture in mobile environments.

2.6.1 SIP Specific Event Notification

RFC 3265 [99] defines an extension to SIP providing an extensible framework by which SIP nodes can request notifications from remote nodes indicating that certain events have occurred. The defined framework has general validity within the SIP architecture; it was designed for providing a framework on top of which simple but nevertheless efficient event based systems could be built. Event types and the way how they are managed by SIP entities must be defined specifically; they are referred to as *event packages*. Section 2.7 describes an example of event package defined for use in SIP.

The general concept is that SIP entities can subscribe to a resource state owned by another entity. The entities that have received a subscription request send notifications when the state of a resource they handle changes, to all the (authorized) entities that have subscribed. Two new SIP methods are defined for the purpose:

- SUBSCRIBE: for subscribing to the services provided by a SIP entity
- NOTIFY: for notifying to subscribed (and possibly authorized) entities when a relevant event occurs at the notifying entity

A typical flow of messages for a general SIP specific event notification scenario is shown in Fig. 2.3, where the entity sending the SUBSCRIBE is referred to as the subscriber, and the other as notifier. These entities are logical end points in the SUBSCRIBE/NOTIFY exchange. The actual flow of messages may, according to SIP logic, follow a chain of SIP proxy servers, as would be the case of an INVITE request.

When a SIP entity wants to subscribe to the services provided by a notifying entity, it builds and sends a SUBSCRIBE message (message M1 in Fig. 2.3). In case of a subscription to the presence event package [100], an example message SUBSCRIBE is:

```
   SUBSCRIBE sip:resource@example.com SIP/2.0
Via:  SIP/2.0/TCP watcherhost.example.com;branch=z9...
To:  <sip:resource@example.com>
From:  <sip:user@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
CSeq:  17766 SUBSCRIBE
Max-Forwards:  70
Event:  presence
Accept:  application/pidf+xml
Contact:  <sip:user@watcherhost.example.com>
Expires:  600
Content-Length:  0
```

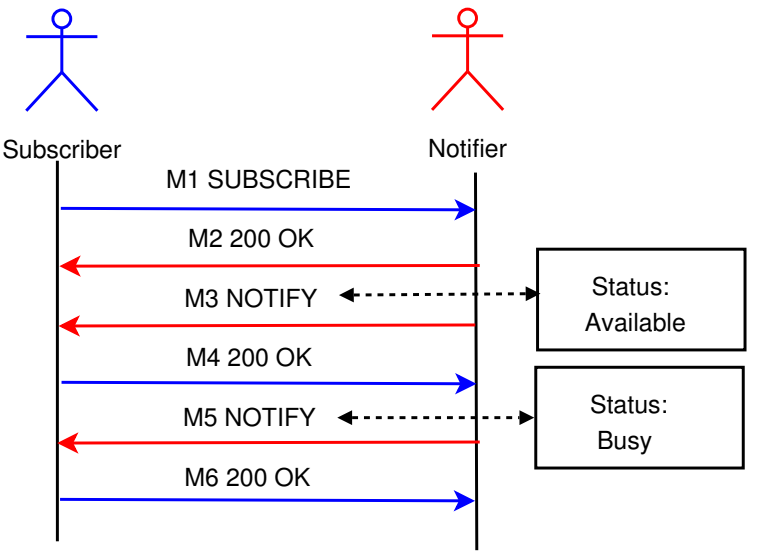


Figure 2.3: General event notification message exchange flow

The message contains in the newly defined Event header field the name of the event package the subscriber is interested in. The Request-URI is the URI of the entity whose event notification the subscriber is interested in. SUBSCRIBE requests create a dialog. Subscriptions have a limited lifetime; SUBSCRIBE requests can contain the subscription duration, as expressed in the Expires header field, in seconds. SUBSCRIBE requests can contain a body where some optional parameters can be provided to the notifier. For example, filter information to specify only the types of event of which a subscriber desires to be notified.

In any case, the definitive duration is set by the notifier and indicated in the 200 OK message (M2) sent if the subscription request is accepted. Duration of subscription is not permanent, but must be refreshed by the subscriber with a SUBSCRIBE request. This is (among other reasons) to avoid to clog the network with useless NOTIFY messages to which the subscriber is no longer interested. The refresh SUBSCRIBE contains a parameter indicating the duration of the new subscription. In order to close a subscription, it is not necessary to wait its expiration, but either party can send a request (SUBSCRIBE or NOTIFY) with a duration parameter value set to zero, which causes the receiving entity to immediate close the subscription.

After the subscription has been accepted, the notifier sends a notification on the state of the subscribed resource or service. This is done by sending a NOTIFY message (M3) on the same dialog as created by the SUBSCRIBE response.

```
NOTIFY sip:user@watcherhost.example.com SIP/2.0
```

```
Via:  SIP/2.0/TCP server.example.com;branch=z9...
From:  <sip:resource@example.com>;tag=ffd2
To:  <sip:user@example.com>;tag=xfg9
Call-ID: 2010@watcherhost.example.com
Event:  presence
Subscription-State:  active;expires=599
Max-Forwards:  70
CSeq:  8775 NOTIFY
Contact:  sip:server.example.com
Content-Type:  application/pidf+xml
Content-Length:  ...

PIDF Document
```

The message, which semantically is a request, contains the description of the subscribed resource/service state in the body, and the event to which the notification refers to. The type of carried body is defined, as usual in all SIP messages, in the Content-Type header field. Such a type must match one of the types that the subscriber has indicated as acceptable in the Accept header field of the SUBSCRIBE message M1. For the presence event package, the notifier reports its complete presence document, describing its presence state. For example, it may report (among other parameters) an initial status of available.

Successful arrival and processing of the NOTIFY request is confirmed by the subscriber with a 200 OK response (M4). Whenever the state of a subscribed resource changes, the notifier will send a notification about the new state of the resource to all its authorized subscribers using a NOTIFY message (M5- M6). For example, if the notifying user switches the status to ”Busy”, a NOTIFY message, containing the whole updated presence document, will be generated. More details on the use of the SUBSCRIBE/NOTIFY framework for presence information notifications are provided later in Section 2.7.1.

2.6.2 The MESSAGE Extension for Instant Messaging

RFC 3428 [16] defines the MESSAGE method, an extension to SIP that allows a stand-alone transfer of instant messages. The transfer is stand-alone as a MESSAGE request does not create a dialog, it can be sent indifferently inside or outside the context of a dialog. Exchange of instant messages using the SIP MESSAGE method closely recalls the functionalities of a pager or short messages (SMS) in cellular telephony. There is no prior explicit association between the endpoints exchanging MESSAGE messages.

The instant messaging (IM) mode defined with the MESSAGE method is opposed to session-mode based IM. In session-mode IM, an association is explicitly

created between the two parties, and the IM session has a clear beginning and termination phase. In SIP, an IM session would be signaled as other session types using the INVITE and other related methods. Session based instant messaging and presence (IMP) extensions to SIP are being defined in the IETF SIMPLE Working Group and are summarized later in this dissertation in Section 2.7.

MESSAGE messages are particular, as they carry at the same time signaling and user data. They are processed exactly as any other SIP message, by intermediary entities along the end-to-end path. It is not advisable to use the MESSAGE method for a long and continued exchange of instant messages, as this would pose high processing load on SIP servers and cause unnecessary congestion. If MESSAGE messages are sent over an unreliable transport protocol like UDP, there is a limit on the size of the body they can carry. Despite such limitations, there may be situations where the use of the MESSAGE method for IM is useful; we utilize it for communication between UA and proxy server residing on the same device. The usage of the MESSAGE message with decentralized SIP is detailed in Chapter 5.

Using this extension is straightforward. In order to send instant messages using MESSAGE, a UA needs to build and send the message to the target user, whose AOR is specified, as usually, in the Target-URI of the MESSAGE request. The message is routed according to SIP operations, and the actual instant message is carried in the body of the message. Since SIP can transparently carry any type of body, provided that the end-points understand the format of data carried, any kind of message can be sent. The most common type is plain text, but for example a JPEG image could be sent along with a MESSAGE message. The message is replied with a 200 OK message, to indicate that the target UA has received the message. The 200 OK response does not however indicate that the human user has read the message, but only that the message has been delivered to destination.

2.6.3 SIP Extensions for NAT Traversal

When one or more Network Address Translators (NAT) separates an end host from the Internet, in most of the cases, native Internet protocols and applications do not work properly and need modifications in order for the NAT to be traversed. SIP is one of these protocols. NAT traversal for SIP is complicated by the fact that SIP can run over reliable or unreliable transport protocols. Moreover, solutions allowing NAT traversal of SIP signaling messages may not solve the problem for the signaled media flows. Finally, the variety of NAT types available makes NAT traversal solutions often bound to the particularly addressed NAT type. In other word, there is not a single solution that can accommodate all the possible scenarios. The problem statement for SIP traversal of NATs is expressed in [11] and summarized below.

When SIP uses UDP as transport protocol, responses to requests are returned to the source address the request came from, and to the port written into the top-most Via header field value of the request. This behavior is not desirable when the client is behind a NAT. RFC 3581 [105] defines a new Via header field parameter, rport, for allowing SIP clients to specify that responses to requests they have issued are returned to the exact IP address and port from which they originated. To explain the situation, we provide an example [105]. A client sends an INVITE to a proxy server which looks like, in part:

```
   INVITE sip:user@example.com SIP/2.0
Via:  SIP/2.0/UDP 10.1.1.1:4540;rport;branch=z9hG4b...
```

The information contained in the Via header field tells that the response to the message must be sent to the port 4540. The sending node is in a network that is behind a NAT. The message is addressed to the node outbound proxy server; the server is in an external network and therefore the message traverses the NAT. The message arrives to the proxy, which notices as source IP address and port, those used by the NAT, say 192.0.0.1:9988. When the proxy tries to forward the response back to the node behind the NAT, it will use, according to SIP specification, the combination 192.0.01:4540, that is, the same address from which the request was received, but the port indicated by the client in the Via header field. However, this makes impossible for the NAT to resolve the real recipient of the message.

The Via header field parameter "rport" has been defined for solving this issue. It is the port equivalent of the "received" parameter used for indicating the IP address from where a SIP request was received. Clients knowing to be behind a NAT can add the rport parameter in the Via header field of the request they send through the NAT. This action triggers the proxy server to write in the first Via header field not only the IP address from which the message was received, as value of the "received" parameter, but also to set the port number as value of the rport parameter. The message that the proxy forwards to its next-hop destination looks partly like (some lines are broken for visual clarity):

```
   INVITE sip:user@example.com SIP/2.0
Via:  SIP/2.0/UDP proxy.example.com;branch=z9hG4bK...
Via:  SIP/2.0/UDP 10.1.1.1:4540;
received=192.0.2.1;rport=9988
```

The message is processed according to baseline SIP operations, until a 200 OK message is routed back to the proxy:

```
   SIP/2.0 200 OK
Via:  SIP/2.0/UDP 10.1.1.1:4540;
received=192.0.2.1;rport=9988
```

The proxy extracts the relevant parameters from the Via header and sends

the response to the address-port pair 192.0.2.1:9988. The NAT/gateway has the mapping for the port 9988 and delivers the message to the intended recipient at the address 10.1.1.1 and at the port 4540. Note that responses must be sent by the proxy from the same port and address on which the originating request was received in order to traverse symmetric NATs.

The solution proposed solves the problem of NAT traversal for outgoing SIP requests, but leaves open the issue of NAT traversal for incoming SIP requests. A solution for allowing incoming requests to be successfully routed through NATs is given by connection reuse [27]. Connection reuse means that incoming requests should reuse the same TCP connection or UDP 5-tuple utilized by a UA when issuing previous outgoing requests. In the SIP case, the reference outgoing request is the registration to a SIP registrar outside the network protected by the NAT. Devices supporting the connection reuse extension must also support the SIP extension for obtaining and using globally routable UA URIs (GRUU) [102]. The connection reuse draft defines which parameters identify a previous outgoing connection, and how it is possible to reuse it. The use of this extension is triggered by the SIP UA behind the firewall; when the UA sends the REGISTER message to its (compliant to the extension) registrar, it adds as parameter of the Contact header field a flow identifier. The registrar registers this extra-information together with the basic contact data. When a request addressed to the UA behind the NAT arrives at the registrar, it will reuse such parameters to refer the flow initiated by the UA during registration phase. At the time of writing, connection reuse is an individual submission, so it is expected to change before being approved as RFC.

NAT traversal for the signaled media must instead be addressed in an application specific fashion. A SIPPING WG ID [11] describes best current practices for NAT traversal for SIP. Different scenarios of media traversal are described, and proposed solutions are listed; due to the variety of media types that SIP can signal (basically, everything), it is very challenging to find a solution that fits all the media types.

2.6.4 SIP Identity

The security methods described in Section 2.2.4 are addressed by the main SIP specification [106]. There is work in progress in the IETF SIP working group for defining enhancements for authenticated identity management [93], where practices and conventions for identifying end users in SIP messages are recommended. According to baseline SIP operations, when a User Agent Server (UAS) receives a SIP request, it is not sure whether the request comes from a legitimate entity or that it has not been tampered with. In other words, it does not have means to verify that the user listed in the SIP header From field is really the entity that has sent the message. This internet draft proposes extensions for SIP to fix this issue.

Referring again to the SIP users introduced in Section 2.5, let's suppose that Alice wants to begin a session with Bob. Alice authenticates with her domain proxy; the specification suggests but does not mandate TLS for first hop authentication. The server, or in general the entity running a so-called authentication service, when it receives Alice's INVITE, checks whether Alice is authorized to populate the From header field with the given AOR. The authentication service then computes a hash over a set of header fields (the From field, among them) and the body of the message, and signs the hash with the certificate for the domain it is responsible for. The hash is inserted in a newly defined SIP header field, "Identity".

The idea behind this approach is that the proxy server, as the entity holding the certificate for its domain and running the authentication service, ensures that the originator of the request is really Alice. This means the AOR listed in the From field has not been tampered with. In general, the authentication service gives means to ensure that the header fields (and the body) of the message that have been hashed have not been modified. The key idea behind the approach is that the SIP message is signed with the certificate of a trusted entity.

When Bob receives the signed message, he processes the hash key and verifies the integrity of the message. A strength of the approach is that Alice's server does not need to append its certificate to the message, so keeping the SIP message small, but provides Bob the address where he can eventually retrieve it, in case Bob does not already have it. Such an address is listed in another newly-defined SIP header field, called Identity-Info.

2.6.5 SIP Certificate Management

The Certificate Management Service [49] is an extension to SIP based on the SUBSCRIBE/NOTIFY framework, allowing SIP User Agents to discover the security certificates of other users. The proposal defines a mechanism for users to exchange their personally signed certificates through a third party trusted entity.

The proposal defines a new event package, namely "certificate". Authorized SIP users who subscribe to this service receive notification about the certificate (or the certificate itself, if they do not possess it) of the notifier. The use of this event package is important when SIP UAs use S/MIME bodies in their end-to-end communication. In some cases, they will need the certificate of a remote user to encrypt the data sent within an S/MIME protected body.

A SIP user who want to publish his certificate to a Certificate Server, must first authenticate to the server, using methods described in the Certificate Management Draft [49], and then publish the certificate sending to the server a SIP PUBLISH request [88] containing the certificate. Once the certificate has been published, the same user can fetch the certificate when accessing from another device. The

most desirable feature of this service is that third parties can fetch the certificate that a user has published into a Certificate Server. For example, if Bob publishes his certificate to a server, Alice can later retrieve it.

The way of requesting the certificate is to send a SUBSCRIBE request to the AOR of the user, whose certificate is queried. The SIP request will be routed until it reaches the SIP server acting as Certificate Server. The server replies to the request with a 200 OK message and then sends the certificate in the body of a NOTIFY message. The NOTIFY message is protected for integrity using the SIP Identity mechanism, described in the previous subsection. If modifications to the certificate occur, for example it expires or is revoked by the user who owns it, a NOTIFY is generated by the Certificate Server.

2.6.6 Registration of Non-Adiacent Contacts

Usually, a REGISTER message is sent directly from a SIP UA to its registrar. However, there are network configurations where a direct connection between UA and its registrar is not possible, as there is one or more SIP entities in the path between them. In such a scenario, the registration scenario described in Section 2.5.1 fails. The most common example of network configuration where there is not direct connection at SIP layer between UA and registrar are 3G networks, as in routing scenario the terminal has to route all of its SIP messages through a specific visited network proxy server, for tracking and billing purposes.

A new SIP extension header field, called Path, has been defined [138] to solve this problem. This header field is used to record the address of the proxies traversed during the registration process. The registrar stores the list when it receives the REGISTER message and provides this information, in complement with the user's bindings, to a querying proxy. The proxy server would not forward an incoming request to the address specified in the Contact header field, as it would happen normally, but to the topmost address in the list contained in the Path header field. Each of the proxy included in the Path list will strip away its address before forwarding the message to the next hop. When the own address is the last in the list, the intermediate proxy forwards the message to the intended recipient.

2.7 SIP-Based Applications

SIP as a stand-alone protocol is usually of little value; SIP in fact provides middleware functionalities that help applications and facilitate communication between users. The communication means are exploited by the applications and the application layer protocols. SIP is transparent to the application used on top of it; after the SIP signaling has completed, the signaled media is exchanged totally indepen-

dently from SIP. This section presents some applications that use SIP as signaling protocol to establish multimedia sessions.

2.7.1 SIMPLE-Based Presence Functionalities

A service of relevant impact in ad-hoc networks is instant messaging and presence (IMP). The IETF SIMPLE working group is defining extensions for SIP to enable support for presence services and session based instant message exchanges. Presence functionalities are enabled by building the Presence Event Package [100], on top of the SIP Specific Event Notification. Instant Messaging (IM) functionalities are carried out by means of a new protocol, the Message Session Relay Protocol (MSRP) [26], described in the next subsection.

The Presence Event Package

When a SIP entity (the subscriber or watcher) wants to subscribe to the presence service of a remote SIP entity it creates a SUBSCRIBE request, carrying the URI of the desired entity (the presentity). The request traverses normally the SIP network (it passes through a chain of proxies as the other requests) until it reaches a SIP presence server, which will generate a response for the SUBSCRIBE request. The presence server, which generates the response is not necessarily the first presence server handling the SUBSCRIBE request; it is also possible that a presence server proxies the request to another presence server, based on local policies decision.

The presence server generating the response (NOTIFY method), referred to as the presence agent (PA) for the presentity, must possess the presence information of the queried presentity. The PA is the logical entity in charge of managing the presence information of a presentity, processing SUBSCRIBE requests, consequently sending responses and notifying to the subscriber changes in the presence status of the presentity.

Upon authentication and authorization of the subscription, a PA sends a NOTIFY message to the subscriber including the presence information and whether the request was authorized. Note that it is possible for the PA to send a ”faked” NOTIFY message, indicating for example that the presentity is off-line when instead the opposite true. Further NOTIFY messages are sent by the PA to all the authorized subscribers when there is a modification of the presence state.

How the presence information is updated at the PA is out of scope of the Presence Event Package, and is application and implementation specific. An XML-based format for expressing presence information, the Presence Information Data Format (PIDF), has been defined [124] to be mandatorily used in the presence event package. Other formats can be used, but all the SIP clients implementing

the presence event package must support the xml-based format.

Partial Notification of Presence Information

According to the baseline Presence Event Package specification, the PA sends to authorized subscribers the whole presence document every time there is a change in the presence status of the presentity. This behavior is inefficient, especially in situations where the subscriber uses a device with low processing capabilities and battery power, or when the access to the network is obtained through slow links, with low bandwidth resources. Transferring the whole presence document, as answer to a change of a single parameter, is under these conditions clearly a limiting factor and it can lead to a perceived low quality of the service advertised by the end user.

One of several solutions to this problem is having a Partial Notification of Presence Information [76]. This ID leverages on the mechanisms proposed in the standard [100] but allows subscribers to request to be notified only about the parameters (called tuples) of the presence information that have changed.

When a watcher subscribes to the presence service of a presentity, it can explicitly request to be partially notified of changes in the presentity's presence state. This is done by specifying the value "application/pidf-diff+xml" [75] as accepted presence information format. This format extends the mandatory PIDF format.

The first NOTIFY message sent to a watcher always contains a complete presence document. Whenever local policy decisions trigger a notification to the subscribed watchers, a PA should send to the watchers that have requested it, only the presence elements that have changed. An incremental version number allows watchers to process and correctly update the received partial presence information. Modifications to the previous presence state, carried in a partial notification, are comprised between the tags:

- *add* If a new element must be added to the presentity presence document
- *replace* If an existing element must be updated
- *remove* If an element must be removed from the presence document

It is up to the implementation to decide how and whether indicate a change in a presentity's presence information to the human user. If a watcher refreshes its subscription by sending a refresh SUBSCRIBE request, a whole presence document must be delivered by the PA. This is done according to the SIP event framework specifications and to cope with situations of network failure and partial notification message losses.

A drawback of the specification for partial notification of presence information is that it mandates the use of the application/pidf-diff+xml presence information format. This format is XML-based and therefore very expensive from the message size point of view. Moreover, presence documents are carried by SIP messages, which are themselves quite big in size. The conjoint big size of XML presence documents and SIP messages can make the NOTIFY message too big to be carried over UDP, as stated in Section 18.1.1 of the SIP specification. The Presence Event Package mandates the PIDF format to be implemented by all compliant devices, but leaves open the possibility to express presence information in other formats. The XML format is instead the only defined for partial notification of presence information.

2.7.2 The Message Session Relay Protocol

The SIP protocol has already been extended with methods specific for sending instant messages [16], as discussed above in Subsection 2.6.2. The MESSAGE extension allows messages to be exchanged outside a session, independently with each other, and it is better suited for short message exchanges. This approach is referred to as page-mode, as it resemble a pager-based exchange of messages. Session-mode messaging, where instant messages are exchanged in the context of a session, presents several advantages, such as explicit rendezvous, tighter integration with other media types, direct client-to-client operation, and increased privacy and security [26]. SIMPLE has defined a new protocol, the Message Session Relay Protocol (MSRP) for session-based instant messages exchanges.

MSRP is a text-based connection-oriented protocol for exchanging arbitrary instant messages, which can be of any format, text or other binary MIME types. MSRP cannot be used as a stand-alone protocol, but only associated with a rendezvous protocol, which independently takes care of establishing an MSRP session. SIP is the protocol used in the specification, but it is not the only option.

MSRP sessions are arranged using SIP in the same way as any other media session is arranged with SIP. When a SIP user invites a remote party to a MSRP session, he sends a SIP INVITE message with an SDP description reporting an MSRP message media session. An MSRP session description contains an MSRP URL indicating where the party wants to receive MSRP requests, that is, instant messages. Similarly to what happens in SIP, every request is answered with a response, in order to complete an MSRP transaction. MSRP defines only two types of requests:

- SEND: for sending instant messages, or chunk of those.
- REPORT: for sending reports on the positive or negative outcome of an MSRP transaction, i.e., request response.

MSRP is carried over TCP, which implies that one of the two parties must open a connection to the other before sending MSRP messages. The connection is opened always by the party sending the offer in the SIP offer/answer session establishment exchange. So, if the initial SIP INVITE has no offer, then the 200 OK in reply will contain an offer and therefore the inviting party is willing to receive a TCP connection. This double possibility allows circumventing firewall limitations, when either one of the parties has incoming TCP connections blocked by the firewall.

An example of MSRP message sent by Alice to Bob is:

```
   MSRP 6aef SEND
To-Path:  msrp:bob.example.net:8145/foo
From-Path:  msrp:alice.example.com:7965/bar
Content-Type:  text/plain

Hi Bob
-------6aef$
```

Alice's request begins with the MSRP start line, which contains a transaction identifier that is also used as a final boundary marker. The URI of the user that is the recipient of the message is indicated in the To-Path header, and Alice's own URL in the From-Path header. These URIs are those that have been negotiated during the SIP/SDP session establishment phase.

The protocol operations described above apply only to peer-to-peer communication, where the parties involved directly communicated with each other. Nevertheless, in some situation MSRP messages need to be relied by intermediate entities. The most common case is when both the endpoints are behind a firewall and neither of them can accept incoming TCP connection. Another example is ad-hoc networks with a gateway node; MSRP communication between a node in the ad-hoc network and an outsider must be relayed on the gateway node. The Internet Draft [15] defines extensions to MSRP for its deployment with relays as well.

When an MSRP client wants to use a relay, it must first authenticate to it. It opens a TLS connection with the relay, therefore authenticating the relay by means of the certificate, and afterwards engages in Digest authentication. Authentication information is carried by the newly defined AUTH MSRP message. Both TLS and Digest authentication are mandated to be used by the MSRP specification.

After successful authentication, the relay provides the MSRP client with an MSRP URI, which will be included in all the requests sent by that MSRP client. A message sent from Alice to Bob through relay example.net, will have in the To-Path field both Alice's and the relay MSRP URI. The message sent by Alice to the relay would look like:

```
   MSRP 6aef SEND
To-Path:  msrp:example.net:9000/aeiug
msrp:bob.example.net:8145/foo
From-Path:  msrps:alice.example.com:7965/bar
Content-Type:  text/plain

Hi Bob
-------6aef$
```

The To-Path line has been broken for visual clarity. Without relays, only Bob's and Alice's MSRP URIs would be present. When the relay forwards the message it moves the leftmost URI from the To-Path header to the leftmost part of the From-Path header. In this way a coherent route is constructed. The leftmost URI in the To-Path header is the next hop to deliver a request or response. The rightmost URI in the To-Path header is the final target.

The MSRP URI of a relay is discovered by the peer that needs its services, usually before initiating a session with the remote party, and added to the SDP session description sent to the other party, following a formatting defined in [15]. The remote party is therefore informed that the session is not peer-to-peer, but a relay is used.

2.7.3 Conferencing Support with SIP

Basic SIP sessions imply the creation of a dialog between the involved parties; the dialog univocally identifies two SIP UAs engaging in a SIP session. This structure makes the extension of the concept of dialog to multiparty communication difficult.

SIP, however, does support several models of multiparty communication. The loosely-coupled model uses multicast media groups; participants to a loosely-coupled conferencing session do not have signaling relationship and there is no central control point. However, a separate control protocol like the RTCP must be used as SIP does not provide specific conference control features. Media streams are sent to a multicast address where all the conference participants have subscribed and routing is taken care of by an underlying multicast routing infrastructure.

An opposite approach is used by the tightly-coupled conferencing model; a SIPPING ID [101] provides an operational framework for using SIP within the tightly-coupled conferencing model. According to the model, all the participants to a conference connect to a central control point, which can also provide additional conferencing services. The central control point, referred to as *focus*, maintains a signaling relationship with every conference participant.

The conference focus is logically a SIP UA and is identified by a URI; par-

ticipants can join the conference by sending an INVITE addressed to the URI of the focus. It is also possible that, based on the conference rules and policies, the focus invites someone to join the conference. A strong requirement in the tightly-coupled model is that the URI can unambiguously identify the focus. The way a participant can obtain the URI of a focus is loosely defined by the framework; SIP or even non-SIP methods can be used to acquire the URI.

The SIP conferencing framework can provide several advanced functionalities; a mixer, controlled by the focus, can be utilized to combine the media streams forming the conference and distributing them to the participants. Media combination and distribution rules are given by the focus.

A conference notification service can be associated to a SIP based tightly-coupled conference session. This service [107] is based on the SIP Specific Event Notification; it allows SIP UAs to subscribe to the ”conference” service to receive notifications of the changes in the state of a tightly-coupled conference. Typically, event triggering notifications are additions or removals of participants to the conference. The entity acting as notifier is the focus. Whenever a join or leave event occurs, the focus will notify the subscribers that the list of participants has changed. Based on the received notification, a user can decide whether to join the conference. It is possible for a subscriber to define a set of filters for telling the focus to send notifications only for events that match the rules expressed by the filter. For example, a list of URI can be provided as a filter, with the rule that a notification is requested only for events concerning those URIs. The filter is carried in the body of a SUBSCRIBE request.

Chapter 3

Ad-Hoc and Proximity Networks

We envisaged that in the future ad-hoc networks will develop and spread. In this section we discuss in more detail general technical implications related to the use of ad-hoc or proximity networks. How to enable session management in a decentralized environment, such as ad-hoc or proximity networks, will be extensively addressed in the second part of this dissertation.

Ad-hoc networks are networks that form spontaneously between nodes in the vicinity of each other. In most cases, the nodes forming an ad-hoc network are mobile, although ad-hoc networks of static nodes may exist as well. Ad-hoc networks formed by mobile nodes, which are those we are interested in, are also called *Mobile Ad-Hoc Networks*, or MANETs.

By proximity networks we mean wireless networks formed by devices that come into radio range and establish a connection. Those networks are generally small in size, and devices are one-hop away from each other. Proximity networks are a particular case of ad-hoc networks. From now on, with the term ad-hoc networks, we will also generally refer to proximity networks. Proximity networks can be considered, from our point of view, as small link-local ad-hoc networks.

A particular type of ad-hoc networks are the so called sensor networks. Sensor networks differ from ad-hoc networks in several aspects, mainly as the nodes are much less powerful, and present lower mobility degree. Usually, sensor networks are formed by several thousands of nodes, as compared to the few hundreds that form the most commonly analyzed ad-hoc networks. The application range of sensor network is in the most widespread case to collect data (for example, temperature, humidity) and communicate them to a sink node, which usually is a more powerful base station. We do not analyze any longer sensor networks as they do not constitute a realistic deployment scenario for our proposed session management framework.

Ad-hoc networks are characterized by a high rate of dynamism, in terms of nodes mobility and topology of the network. Nodes can leave or join the network

at any moment, or change their position in the network with a relatively high rate. The effect of this mobility is that the topology of an ad-hoc network is highly variable, both in terms of number of nodes that form the network and in the interconnection among them.

The dynamic topology mainly affects the routing algorithm of ad-hoc networks; routing is probably one of the most challenging ad-hoc research issues. Routing is not straightforward as in ad-hoc networks it is not possible to rely on stable routes connecting the nodes. Instead, node failures or voluntary leaves, or changes in their position, cause frequent disruptions in the discovered routes. This implies that routing paths must be restored, or new paths must be discovered, involving routing messages to be sent over the air and consequent resource consumption.

Since MANETs are formed spontaneously, all the services in the network must be provided by the participating nodes. In infrastructured networks, the client-server paradigm dominates the service models; well-known entities are always available, and nodes can request a service knowing that the remote server will reply. In ad-hoc networks, nodes do not usually know what services are available, and the addresses of the nodes that provide the desired services. A service discovery protocol must be used to find out about services availability in the network.

Besides service discovery issues, load balancing considerations are extremely important in ad-hoc networks. It is not feasible to concentrate high amount of computation and message processing on a little amount of nodes. Ad-hoc nodes are powered by batteries, which would quickly drain if a node would have to continuously perform operations. For example, an inefficient routing protocol may build its routes so that most of the traffic is concentrated through a small subset of nodes. This should be avoided, especially if we consider that routing operations do not directly provide a service to the user of the node that is routing a message, and contribute to reduce the device battery life. In the rest of the chapter, we expand the issues related to ad-hoc networking, providing reference to related work[1]. We also present future deployment scenarios for ad-hoc and proximity networks.

3.1 Physical and Link Layer Technologies

Currently, the two technologies that mostly are deployed to build ad-hoc networks, are Bluetooth and WLAN, or IEEE 802.11. Ferro and Potortì [33] provide an

[1] An extensive and frequently updated list of ad-hoc networks related articles and documents, divided by topic, can be found in the web page `http://w3.antd.nist.gov/wctg/manet/manet_bibliog.html`, accessed October 9, 2007

overview of these two technologies and compare their main features.

Both Bluetooth and 802.11 are standards for wireless communication within short range and with relatively low power consumption so that also battery powered devices can use them without draining the batteries too fast. However, they focus different types of devices and connectivity purposes. Bluetooth aims at substituting cables and interconnect mostly small devices, within a very short range, in cells of less than ten meters of range. The IEEE 802.11 technology is the wireless replacement of the Ethernet-based LANs; this is a reason why the IEEE 802.11 technology is also referred to as WLAN. WLAN guarantees higher data speeds and cell coverage area than Bluetooth. Bluetooth ensures data rates of up to 1 Mb/sec. The IEEE 802.11 a and g standards can reach up to 54 Mbit/sec.

The price to pay for the better performance of IEEE 802.11 is battery consumption. Bluetooth is optimized for being used by small devices, e.g. PDAs or mobile phones, with low batteries capabilities in a short range; transmission power can therefore be reduced. IEEE 802.11 is designed for covering a longer-range and being used by bigger devices that can support higher power consumption rates, typically laptops. The current absorbed by IEEE 802.11 is in the average range of 100-350 mA; on the contrary, Bluetooth utilizes a hundred time less energy [33]. Optimizing and reducing energy consumption requirements of IEEE 802.11 is an active research topic and one of the key success factors of widespread deployment of this technology in future portable hand-held devices.

Both technologies operate in the same unlicensed 2.4 Ghz frequency range[2], which makes interferences raise when the two technologies are employed in the same area. On the other hand, interference is an issue also within the single technology, if the number of users is high, since the wireless medium is shared.

The two technologies allow for different network sizes; the basic Bluetooth building block is a *piconet*, where up to 8 devices can be connected at the same time. IEEE 802.11 basic blocks are cells called *Basic Service Set (BSS)*; a single BSS can support up to 2007 nodes [33] when an access point is used. In ad-hoc mode there are no theoretic limits in the number of connected devices in a single BSS. Clearly, since these nodes would share the same medium, it is not practical to configure a WLAN network with so many nodes per BSS. The range of a Bluetooth piconet is 10 meters, while a WLAN BSS can cover up to 100 meters.

Both of these two technologies as such are currently inadequate to enable efficient ad-hoc communication. Bluetooth is energy-friendly, but does not provide high enough data speeds and coverage range to support ubiquitous communication. On the other hand, WLAN has the opposite characteristics of reasonable data speeds and coverage range, but unsuitability of deployment in power constrained

[2]IEEE 802.11a operates in the 5Ghz range.

devices. A new version of Bluetooth has been released in November 2004[3]. By employing a different modulation scheme for the payload data, it can reach up to 3 Mbit/sec speeds.

Other wireless technologies are becoming focus of interest from research and industrial world. The IEEE is standardizing the 802.16 Wireless Broadband Access (BWA) technology[4], meant for Metropolitan Area Networks. This standard aims at extending fiber optic networks providing support for high speed data without the need of building a costly infrastructure. The technology is a wireless substitute for last mile technologies, like cable or ADSL. It is not specifically meant for building ad-hoc networks, but it is a good example of a possible future wireless network, where distributed session management can be applied due to the relatively high data rate ensured to devices interconnected with this technology. In any case, connecting devices with IEEE 802.16 as such, presupposes the presence of an infrastructure, i.e., the antenna. However, IEEE 802.16 may be used in conjunction with other wireless (possibly ad-hoc) technologies, having lower radio range, to provide more advanced services, as in the case of 4G networks. We discuss examples of interaction of ad-hoc and infrastructured networks later in this chapter, in Section 3.9.2. The IEEE 802.16 technology is also known with the name of WiMax, or Worldwide Interoperability for Microwave Access.

Another IEEE working group (WG), 802.15[5], addresses wireless technologies meant for a shorter range than WiMax, namely Wireless Personal Area Networks (WPAN). This WG is divided into several task groups. Of them, the 802.15.3 WG is chartered to design a standard for high-rate (20Mbit/s or greater) WPANs, with a range of few dozens of meters. The 802.15.3 standard is strictly connected to the Ultra Wide Band (UWB) technology, which doesn't identify a definite standard of wireless communication but it is rather a method of modulation and data transmission. Ad-hoc peer-to-peer networking is explicitly addressed as a feature of this standard. The solution seems a promising technology for the future ad-hoc networking needs. A major problem in its widespread development may be that it is very difficult to detect and therefore difficult to regulate, due to its low power requirements.

The task group 4 activities have gained noticeable interest, and produced a standard known with the name of IEEE 802.15.4 or ZigBee. ZigBee promises to

[3]The specification can be downloaded by registered users in `https://www.bluetooth.org/spec/`. Site accessed on October 9, 2007. The web site `http://www.radio-electronics.com/info/wireless/bluetooth/bluetooth_edr.php`, accessed October 9, 2007, provides a summary of the new Bluetooth specification

[4]The IEEE 802.16 Working Group on Broadband Wireless Access Standards, `http://grouper.ieee.org/groups/802/16/`, accessed October 9, 2007

[5]IEEE 802.15 Working Group for WPAN, `http://grouper.ieee.org/groups/802/15/`, accessed October 9, 2007

be a low data rate solution targeted for ensuring a prolonged (from month to years) battery life and very low complexity. It is operating in an unlicensed, international frequency band. Potential applications for ZigBee are sensors networks.

Wireless USB[6] is a new technology created within the USB consortium to remove the need of cables from USB connections. It promises data speeds of 480Mbit/sec at 3 meters range and of 110Mbit/sed at 10 meters. Wireless USB uses the UWB technology at physical layer; it allows interconnecting up to 127 devices to a single host [40]. Due to the presence of a centralized host device in the network architecture, Wireless USB is a candidate for the replacement of Bluetooth in the future, interconnecting devices in an office or at home. Since the host is the central point of the architecture, the usage of Wireless USB as a peer-to-peer ad-hoc network is not immediate. All the wireless traffic is relied through the host, which would make its batteries drain too quickly. The best application of wireless USB is achieved when the host is a machine without resource limitations. For example, wireless USB can be used for connecting home devices, and the host could be a computer controlling other peripheral devices.

3.2 Establishing IP Connectivity in Ad-Hoc Networks

Bluetooth and IEEE 802.11 ensure connectivity at link layer. However, they do not provide higher layer connectivity, and need external addressing and routing mechanisms in order to ensure global connectivity among the stations [33]. It is expected that ad-hoc networks will be IP-based. Therefore, once link-layer connectivity is gained, ad-hoc nodes need to acquire an IP address valid at least locally in the ad-hoc network. This is not a straightforward procedure, due to the lack of centralized entities managing the assignment of IP addresses. In isolated ad-hoc networks, IP address distribution must be performed in decentralized way, which may lead to address duplications.

Several proposals discuss how to acquire IP addresses in ad-hoc networks, both for IPv4 and IPv6. A recent algorithm for address autoconfiguration for IPv6 in ad-hoc networks, and a review of other proposals, can be found in [30]. The authors divide the autoconfiguration mechanisms into three categories:

1. Stateless address autoconfiguration-based
2. Specific distributed systems algorithms-based
3. DHCP-based

[6]Wireless USB, `http://www.usb.org/developers/wusb/`, accessed October 9, 2007

A simple solution for address autoconfiguration for IPv4 is presented in RFC 3927 [18]. According to this proposal, the addresses to be used in a link-local environment are randomly chosen from the address space 169.254/16. Mechanisms for detecting and managing conflicts in address assignment are defined in the RFC. This mechanism can be straightforwardly applied to link-local ad-hoc networks, that is, proximity networks. One of the first proposals for IP address autoconfiguration in ad-hoc networks is given by an expired Internet Draft [91] by Perkins et al. Addresses are randomly chosen from the network space 169.254/16 in case of IPv4, or on prefix MANET-PREFIX in case of IPv6. In IPv4, an ad-hoc node temporarily picks a random address from the available pool, and floods an Address Request message. If no reply is received, the node assumes that the address is unique and uses it permanently. This approach does not allow conflict detection in case two partitioned ad-hoc networks merge, as address detection is performed only once when joining the ad-hoc network.

IPv6 stateless address autoconfiguration is described in RFC 2462 [128]. The scheme proposed is general; Weniger and Zitterbart [136] have modified it for providing IPv6 stateless address autoconfiguration in large scale ad-hoc networks. An ad-hoc node that needs to acquire IPv6 connectivity, first performs the steps described in RFC 2462 for acquiring an IP address, and then executes duplicate address detection (DAD) procedures. A hierarchical structure is built in [136] to reduce the overhead due to DAD procedures.

Nesargi's and Prakash's proposal [86] belongs to the second group. Their solution allows nodes joining an isolated ad-hoc network to acquire IP connectivity relaying the responsibility of assigning the IP address to the nodes that already form the network. Particularly, the newcomer selects one of its neighbors as the initiator node. The initiator node will choose an IP address from a pool of available addresses and takes care of detecting duplications. If the selected address is usable, e.g. it is not duplicated, it is returned to the newcomer.

A solution belonging to the third class is described in the paper [83], where a new protocol for autoconfiguration, the Dynamic Configuration Distribution Protocol (DCDP) is defined. DCDP extends DHCP to a stateless autoconfiguration protocol for wireless and wired networks. Specific DCDP entities are in charge of assigning the addresses; address assignment is done by splitting in two the address pool assigned to a DCDP entity and passing it to down in the hierarchy to the requesting entity. The partitioning rule is simple, as nodes do not need to keep state information about assigned addresses, like proposed in the paper [86], but presents scalability problems, as it may lead to several unused IP addresses.

In small scale ad-hoc networks, stateless autoconfiguration approaches are more suitable; they can handle efficiently autoconfiguration in link-local environments. The relevant problem of these mechanisms is that they do not respond

well to events like network partitioning. The hierarchical approach presented in the paper [86] is more suited for bigger networks. Its disadvantages lie in the fact that the keeping of state information and the choice of an initiator node make the algorithm complex.

The proposal by Fran and Subramani [30] targets small-medium scaled ad-hoc networks; it is based on the IPv6 Stateless Address Autoconfiguration Protocol [128] and the Neighbor Discovery protocol [85]. It addresses the problem of network partitioning by allowing the neighbor solicitation messages to have a site-local scope, rather than link-local.

3.3 Acquiring Basic Services in Ad-Hoc Networks

Ad-hoc networks can be deployed for any purpose: the kind of applications that will run will determine the type of services needed. For example, if the ad-hoc network is created in a corporate building, a required service may be ”print”. Nevertheless, there are some basic services that are useful regardless of the purpose of the ad-hoc network. These services are Domain Name System (DNS) and gateway to infrastructured networks.

Jeong et al. [50] propose a name service system based on secure multicast for IPv6 mobile ad hoc networks. The basic idea is that each mobile node runs functions of Responder and Resolver of ad-hoc DNS queries. Security is achieved by authenticating the exchanged DNS messages using the so called *Secret Key Transaction Authentication* [133]. DNS queries are multicasted to a well-known address; all the nodes registered to the address receive the message and the responder holding the queried record returns an unicast reply to the resolver. This approach is extremely similar to our method of distributing SIP registration messages in ad-hoc networks, described later in Chapter 5.

Another paper [28] proposes to embed name resolution requests and related replies into the routing messages exchanged to build paths in the ad-hoc network. The paper also presents how to integrate the DNS service in ad-hoc networks with external networks. A gateway node, with connectivity to the Internet, will act as proxy and forward DNS queries to external DNS servers. DNS queries and replies referring to external sources are as well exchanged within routing messages.

A paper from Xi and Bettstetter [139] summarizes the discussion carried out in this and the previous subsection. The paper addresses issues on gateway discovery, IPv6 address autoconfiguration of mobile ad-hoc devices, and the routing and addressing of messages to the gateway. The gateway is logically treated as the access router to the infrastructured network.

Xi and Bettstetter analyze the same scenario when there is no direct connectivity between the access router and the mobile node. In other words, they analyze

scenarios of ad-hoc networks connected to the Internet through a gateway node. The gateway has a double protocol stack; on the Internet side, the standard TCP/IP stack is implemented. On the ad-hoc network side, an ad-hoc routing algorithm is implemented at network layer; link layer and physical layer functionalities depend on the kind of wireless connectivity to the ad-hoc network, i.e., Bluetooth or IEEE 802.11. Communication between ad-hoc nodes and Internet nodes is managed using Mobile IPv6 [51]; moreover, various schemes for routing messages among ad-hoc nodes are presented. This allows building a complete framework for communicating within the ad-hoc network and with external infrastructured networks.

3.4 Routing in (Mobile) Ad-Hoc Networks

Routing is one of the hottest topic in ad-hoc networking, especially in mobile networks (MANETs), which are probably the most common scenario of ad-hoc networks. The amount of research dedicated to this single aspect is impressive; it is not hazardous to say that routing is the single ad-hoc networks related topic where the highest number of papers has been produced. The main reason for this is that ad-hoc routing is not easy, as ad-hoc networks are highly dynamic being formed by nodes that can join or leave at a high rate. Moreover, in some cases a node departure is not graceful, but due to failures like loss of connectivity or draining of batteries.

These factors contribute to the fact that the topology of ad-hoc networks is highly instable, and this affects the routing algorithms, which must cope with frequent variations of the network topology. It is also worth to add that most often, ad-hoc nodes are small devices, with limited capabilities; therefore, ad-hoc routing algorithms cannot be too complex or require to keep too much state information. Traditional Internet routing protocols cannot be applied for ad-hoc networks. They in fact assume a more stable network topology than ad-hoc networks and are meant to be run in powerful dedicated machines.

The IETF MANET working group[7] is actively working on the topic of routing in mobile ad-hoc networks. Its main tasks are of defining two classes of routing protocols. One uses a reactive approach, while the other the complementary proactive approach. We will now briefly discuss the main differences between proactive and reactive routing, mentioning the main protocol defined in MANET for each class. The discussion is meant to give an example of the approach in ad-hoc routing; a thorough discussion on other proposed routing protocols is out of scope of this dissertation. This classification is not the only possible; routing

[7]IETF MANET Working Group: `http://www.ietf.org/html.charters/manet-charter.html`, accessed October 9, 2007

protocols can also be classified based on their addressing scheme, that is, whether it is flat or divides the network into several hierarchy levels. Other protocols, take energy-awareness as central factor, and try to perform routing maximizing energy saving. A thorough review, classification and comparison of several unicast ad-hoc routing protocols is presented in the journal paper [2].

3.4.1 Proactive Routing Protocols

The common denominator of proactive routing protocols is that each node maintains routing information for other nodes in the network. Depending on the protocol, the portion of network for which nodes keep routing information varies. Routing information is updated every time a particular event, like change in network topology, occurs; alternatively, there can be periodic updates.

An example of proactive routing protocol developed by MANET is the Optimized Link State Routing Protocol (OLSR), specified in the RFC 3626 [21]. OLSR is a point-to-point routing protocol, based on the link state algorithm developed for Internet routing. With OLSR, ad-hoc nodes maintain routing information by means of periodic exchanges of messages. Control messages needed to periodically spread information on the topology of the network are broadcasted. However, only a subset of a node's neighbors, referred to as the Multipoint Relays (MPR), will forward the information coming from a given node. The other neighbors can process and use it, but they do not forward the message. The policy with which a node selects its set of MPRs, is that the MPR set must cover all of the node's 2-hop neighbors.

The inherent advantage of OLSR, compared to other routing proactive protocols, is that only a subset of neighbors forward routing control messages for a given node. This greatly reduces bandwidth consumption and increases scalability. However, it requires nodes to have and maintain knowledge on all their possible 2-hop neighbors. In big networks this may be complex and require to keep lots of information. Frequent variations in network topology also add complexity to the overall algorithm. Another IETF proposal of proactive ad-hoc routing protocol is Topology Dissemination Based on Reverse-Path Forwarding (TBRPF) [89]. TBRPF is a proactive, link-state routing protocol designed for mobile ad-hoc networks, providing hop-by-hop routing using the shortest path metric for each destination.

3.4.2 Reactive Routing Protocols

Reactive (or on-demand) routing protocols try to build a route to a destination when it is needed, that is, when an ad-hoc node tries to send a message to the destination for which it does not have a valid route. The general approach used

by reactive routing protocols is to flood a route request message through the network, until a node with a valid route to destination, or the destination itself, is found. That node will reply to the route request. There is no need to periodically broadcast route maintenance information.

Reactive protocols can be further classified into two sub-categories: source routing and hop-by-hop protocols. In source routing protocols, messages carry in the header the complete path to destination. Intermediary nodes forward the message to the next hop and do not need to keep routing information for every active node in the network. The assumption is that the source node has previously discovered the path to destination, using protocol-specific methods. The MANET WG is developing a reactive source routing protocol, the Dynamic Source Routing (DSR) [52].

A clear disadvantage of this approach is that in large networks, the header can grow considerably and consume too much bandwidth; this affects the scalability of the protocol. Additionally, in large networks, the probability that an intermediate path to destination gets broken is higher. However, since a source can discover more than one path to destination, source routing protocols may be more robust than hop-by-hop ones if one of the discovered path fails. Moreover, the lack of periodic broadcast control messages allows nodes to save battery resources, i.e., entering into sleep mode, when they are not active. In any case, if network topology is highly dynamic, this approach is not suitable, as a route to destination may change while the discovery process is still going on.

In hop-by-hop routing, each routing message contains only the information needed to contact the next hop to destination. Each intermediate node in the end-to-end path consults a routing table to forward a message towards the recipient node. This approach can adapt better than the other reactive one to dynamic environments, as the choice of the best path is done hop-by-hop and by an immediate neighbor. The price to pay is the fact that nodes must maintain and update information on the topology of the network and must be aware of the condition of their neighbors.

The reactive hop-by-hop ad-hoc routing protocol developed in the MANET WG is called Ad-Hoc On-Demand Distance Vector Routing (AODV) [90], and it is specified in the RFC 3561. The route discovery strategy performed in AODV is very similar to DSR. When a node needs to find a route to an unknown destination, it sends a route request message. Route requests in AODV are flooded in the network; each node that forwards a route request diminishes a hop-counter in order to stop the flooding at some point. A unicast Route Reply message is returned by the node that knows a route to destination. The difference with DSR is that messages in AODV contain only the IP address of the destination node, instead of complete route information. Route request and route reply messages in

DSR, carry the address of each hop en route to destination.

The MANET RFCs describing the above mentioned routing protocols are experimental. The next steps will be of standardizing a proactive and a reactive routing protocol, and a multicast routing protocol, also utilizing the experiences and knowledge gained with the design of the experimental protocols.

3.4.3 Discussion

It is not possible to state that proactive routing protocols perform better than reactive ones or vice versa. Their performance is related to several factors and heavily depends on the network topology and characteristics. In general, proactive protocols have the advantage that routes are immediately available as their discovery is performed in advance. Reactive protocols, on the other hand, do not offer this possibility, as new routes are discovered on-demand. Time-sensitive applications may suffer from delays in route establishment of reactive protocols.

However, proactive protocols periodically flood the network to maintain a complete set of routes, and are therefore more bandwidth aggressive than reactive protocols, which flood the network only if the need arises. However, nodes activity plays an important role on the bandwidth consumption, and influence the routing overhead due to reactive routing protocols. In dynamic environments, proactive discovery of routes may be detrimental, as network topology varies so quickly that the proactively discovered route information becomes soon obsolete.

The size of the network is also an issue; in (reactive) source routing protocols the fact that the destination is several hops away increases the size of the headers and makes control overhead relevant. However, from the routing point of view, when referring to the size of the network, the addressing scheme used in the network is more important than the time when route discovery is performed. If the addressing scheme is flat, then it may not scale well to big networks. For such networks a hierarchical approach, where, for example, some nodes are chosen as cluster heads and are in charge of forwarding messages from all the nodes in their cluster, may be more suitable. The biggest problem in hierarchical cluster-based approaches is the complexity of the algorithm and the necessity of electing suitable cluster heads and coping with their unannounced failures.

A common characteristic of most of the ad-hoc routing protocols that have been proposed so far in research papers is that, for sake of generality, they are applied to very big ad-hoc networks, with several hundreds or sometimes even thousands of nodes. Tschudin et al. feel that this is not the right approch to achieve wide deployment of ad-hoc networks in real life [129]. Their claim is that, although simulation-based research on ad-hoc networks has produced noticeable results, there is the need to carry out experiments addressing realistic MANETs deployment scenarios. Such realistic scenarios are deemed to be small-scaled ad-

hoc networks, with few dozens of nodes (typically, around 20) at most, two or three hops away from each other.

We agree with this idea of a realistic target environment; the vision of future ad-hoc enabled communication environments presented in Section 1.1 is enabled by small ad-hoc networks. Such networks have seldom more than a couple of dozens of nodes; in some scenarios, the nodes can be even all directly connected with each other at link layer. We believe that exploiting link-local environments, that is, proximity networks, must be carefully considered. Few dozens of nodes directly connected at link layer would form realistic ad-hoc networks, where nodes are relieved from routing operations, suffering therefore from less power consumption. Increased batteries lives and bandwidth availability due to the absence of routing control messages, can be powerful triggers for commercialization of ad-hoc networking into the mass market. Nevertheless, the session management framework presented later in the second part of this dissertation has been designed for small link-local proximity networks and bigger ad-hoc networks, with nodes several hops away from each other.

3.5 Service Discovery

Service discovery is probably the higher layer topic that has gained the biggest interest among researchers, with reference to ad-hoc networking. Enabling service discovery in ad-hoc networks means giving the nodes the possibility to find out which services are available in the network. Since ad-hoc networks are formed as spontaneous aggregation of peer devices, often it is not possible to know which services are available, or who provides them. A good example of service that one or more ad-hoc nodes could provide is gateway to an external network, such as the Internet.

Service discovery is not exclusive to ad-hoc networks, but can be employed in infrastructured networks as well. In such environments, a typical configuration is that a centralized server holds information about the available services; a client-server paradigm is used, and the service discovery protocol carries queries to the server and its replies back to the querying node. In ad-hoc networks such approach is clearly unfeasible, and search for services must be done in a distributed way. A desirable property for a service discovery protocol in MANETs, is to allow discovery of services not only from the neighbors of the node, but also from nodes several hops away.

Service discovery protocols can be categorized in pull and push protocols. In the pull approach, nodes pull the desired information whenever it is needed, either from a centralized entity or from the network, using distributed algorithms. In push protocols, nodes who offer a service, or a centralized service provider,

publish (push) the information on the service to the network whenever it comes available, its characteristics change, or periodically.

A lot of research has been conducted to provide solutions for service discovery in ad-hoc networks. In most cases, it consists of individual proposals addressing only a specific type of ad-hoc environment. Providing a taxonomy of all existing proposals is out of scope of this dissertation. Instead, we now briefly introduce some of the major service discovery frameworks, proposed by industrial or academic consortia, and comment on their suitability in ad-hoc networks.

Jini [125, 126] is a Java-based proposal from Sun Microsystems that utilizes the centralized pull approach. The strength of Jini is the support for very dynamic and flexible distributed systems, in which participating devices can join and leave without any need of administrative work. Such flexibility is a desirable property in ad-hoc networks; however, the centralized approach makes Jini unfeasible for use in ad-hoc networks.

Universal Plug and Play (UPnP) [35] is a proposal defined by the UPnP forum, which is led by Microsoft. It utilizes a distributed pull approach on top of HTTP. The distributed approach is suitable for ad-hoc networks; however, service features and capabilities are described with XML, and such a format may be too bandwidth aggressive in MANETs. Moreover, it supports only known devices [127], which makes it scarcely flexible.

Web Services Dynamic Discovery [22] is a multicast discovery protocol for locating Web services. It allows discovery of services in ad hoc networks that have a minimum of networking services (e.g., DNS or directory services). The Web Services Dynamic Discovery protocol has a simple architecture, which suits well ad-hoc networks since it uses multicast thus allowing only nodes registered to the web services multicast address to receive queries. When a node joins the network, it may send a join message to the multicast group to inform the other participants. The main drawback is that web services needs another protocol to use the discovered services and cannot be utilized stand-alone and also that the approach is not effective in totally isolated ad-hoc networks.

Bluetooth Service Discovery Protocol [44] is a protocol designed to solve service discovery problem between Bluetooth enabled devices. It does not provide access to services, brokering of services, service advertisements, or service registration and there is no event notification either. SDP supports searching by service class, search by service attributes and service browsing. Methods of invoking the found services are outside the scope of the protocol. This solution is tied to the Bluetooth technology, so it may be not efficiently portable to other environments, like WLAN.

The Service Location Protocol (SLP) [41] is the IETF proposal for service discovery. It provides fully decentralized operations and scales from small, unad-

ministered networks to large networks. It enables security policies stating which users are allowed to discover which resources. Another key functionality of SLP is its easiness of configuration, which allows its deployment without excessive administrative efforts. SLP can be deployed in ad-hoc networks without major modifications, by utilizing only a subset of its architectural functional entities.

SLP defines Directory Agents, which are centralized entities in charge of storing and distributing service information. Such entities are mainly important for guaranteeing scalability of SLP networks. Service discovery can also be performed in a decentralized fashion, by having each SLP node implement a User Agent and a Service Agent. The user agent is the logical client entity in SLP, and performs queries for SLP services. Such queries are replied to by the service agents. When a Directory Agent is present, Service Agents register their services to the Directory Agent. User Agents address their service queries to the Discovery Agent. We have used SLP in our decentralized session management framework as an alternative to SIP for discovering SIP users in the network. We present the interworking between SIP and SLP later in Section 5.9.

3.6 Instant Messaging and Presence in Ad-Hoc Networks

Defining middleware and lower layer services for ad-hoc networks is a necessary step towards the final goal of building end user applications. It is of little use to design, e.g., an efficient routing protocol for MANETs if eventually there is no application that can use it. Implementing usable applications for ad-hoc networks is therefore necessary to guarantee a widespread success of this technology.

Clearly, the decentralized and dynamic nature of MANETs must be reflected also at application level, by optimizing applications for use in this particular environment. It is not possible to simply take applications designed for the Internet and deploy them in ad-hoc networks. We now describe some proposals for adapting instant messaging and presence (IMP) services to ad-hoc networks.

A solution, not meant for allowing exchange of instant messages among the nodes of an ad-hoc networks, but consisting rather in a notification service is provided in [141]. Users in hot spots can aggregate and form dynamic groups, where one group member acts as a proxy to the Internet in charge of collecting from existing Internet IMP service notification of messages and broadcast them to the other group members. The interested members can then access directly the server when they know that messages are available for them. This approach allows saving battery resources, since nodes do not need to stay connected all the time and wait for messages, as this responsibility is delegated to the proxy. Since the proxy node changes periodically, there is no risk that a single node drains its batteries to serve the other group members. A specific protocol has been

implemented to collect notifications from Internet servers and broadcast them to the group members.

The paper [39] defines DAWN, a decentralized IMP system for small scaled ad-hoc networks, which allows users to chat and use presence services in MANETs. The chosen approach is very similar to ours (presented in the second part of this dissertation), as they move the centralized server operations to all the ad-hoc nodes, letting them manage the presence information themselves. The protocol chosen for the exchange of messages is the Extensible Messaging and Presence Protocol (XMPP) [109, 110], commonly known as Jabber. Presence information from other nodes of the network is retrieved by means of periodical polling; support for one-to-one messaging and chat rooms is provided as well. Although the system is based on a standard IMP protocol, it does not aim at being interoperable with it, and moreover, it was designed while XMPP was being standardized, so that DAWN may no longer be interoperable with the current XMPP protocol.

Favela et al. [31] propose AIDA, an extended IM architecture for supporting spontaneous interactions in ad-hoc networks. AIDA uses a Jabber server to support notifications for availability of services in ad-hoc networks. In other words, AIDA is a service discovery architecture built my means of an instant messaging protocol. However, AIDA is not entirely based on Jabber, but also introduces some proprietary elements, which makes it a stand-alone architecture, although it is based on a standard protocol.

The article [130] describes WACNET, a framework for allowing off-line and on-line messaging services in ad-hoc networks, using proxy nodes. Off-line messaging comprises e-mails, or instant messages sent to off-line users, while for on-line messaging is meant real-time interactive communication with users that are connected to the network. The idea in WACNET is of defining a set of nodes (proxies) that take care of forwarding a message to a given peer, when the sender does not have the possibility of delivering the message itself, e.g., because the node is leaving the ad-hoc network. A proprietary routing protocol is also defined in order to correctly forward messages to the intended recipients.

3.7 Security Considerations

Security enforcement in ad-hoc networks is a particularly challenging task. The reason is that the security issues referring to infrastructured network apply to ad-hoc networks as well, but in MANETs, they are complicated by the fact that MANETs are collection of nodes that usually do not have any pre-existing security association with each other. Ad-hoc nodes do not have ways to verify the credentials of a new user, especially if the ad-hoc network is isolated from external networks. The lack of infrastructure implies the lack of a centralized and

trusted entity, which can verify the security credentials presented by a node. This consideration translates into the fact that, in MANETs, mutual trust relationships between nodes are often necessary, and that maintaining a certain degree of trust is unavoidable.

Routing protocols are mostly based on the consideration that intermediate nodes do not misbehave and forward packets according to the algorithm. Secure operation of routing protocols is anyway a fertile ad-hoc network research area; in general, security is seen as an extension to the original operations of the protocol. When the assumption that the intermediate nodes do not misbehave is not true, countermeasures should be taken to avoid segmentation of the network. Segmentation means that some nodes are cut out by malicious nodes that dump the packets they receive instead of forwarding them. Segmentation actually results in a Denial of Service (DoS) attack, as an ad-hoc node is prevented from accessing the network, and causes unnecessary power consumption, due to the node attempts to deliver the data to destination.

Selfish behavior is another cause of problems in ad-hoc networks. The motivation behind it is that ad-hoc nodes often rely on batteries, which are a limited source of power; in some cases, the data that a node is processing are not meant for the node itself, but rather are third party packets that the node has to forward. Receiving a packet, processing it, running a routing algorithm, and eventually transmitting the packet to the proper next hop consumes resources and does not give any direct advantage to the user.[8] Therefore, users may configure their ad-hoc devices in a way that all the routing traffic of third parties is dumped. This is a form of denial of service in the ad-hoc network, and closely recalls the phenomenon of free riders in peer-to-peer systems.

When devices rely on a limited source of power like those forming ad-hoc networks, another type of denial of service attack is possible. A malicious node can decide to shut down a victim node by sending it unwanted data at high pace, in order to quickly drain its batteries. This attack is clearly more effective if more attackers target a single victim.

The nature itself of the communication medium in MANETs is a source of security attacks; wireless links are easily exposed to eavesdropping attacks. For such a reason, it is desirable that sensible information be protected for confidentiality, for example, by means of encryption. Encryption poses several issues. It is necessary that both the sender and the receiver share a security agreement in order to encrypt and decrypt messages. Similar considerations apply also if integrity protection is needed. Encryption algorithms may be resource consuming, and not easily sustainable by small devices. Confidentiality should therefore be used only

[8] According to the routing protocol used, the advantage may be an update of the routing tables. But this is not something that the human user notices directly.

when necessarily needed, at least when smaller devices are employed.

F. Stajano and R. Anderson have listed [122] types of security attacks to ad-hoc networks, which are not strictly related to routing. Such attacks include denial of service, authentication, and naming; the challenges when facing such attacks are that their nature and the countermeasures for coping with them are different than in the conventional network environments. Similar issues are addressed by Hubaux et al. [46] in a paper providing an overview of security problems for MANETs and identifying the treats to the conventional security mechanisms, when applied to ad-hoc networks.

3.8 Mobile Networks in the Near Future

The maximum theoretic bandwidth availability guaranteed to a single user by 3G systems is 2 Mbits/sec[9], in situations of low or no mobility and absence of other users competing for radio resources. In practice, such a link speed will never be ensured, and can even be insufficient in environments where the concentration of users is high. This is one of the reasons why a paradigm different from the cellular wireless networks has been subject of interest in the research world in the past few years. This is also a reason why, while 3G systems are still far away from being commonly widespread in the mass market, the word 4G is buzzing in the ears of academics and industrials.

4G networks are defined as networks capable of providing high data speed and support global roaming across multiple wireless and mobile networks [131]. Among the multiple wireless networks that 4G systems promise to support, WLAN has attained particular attention. WLAN systems introduced in hot-spots can provide data speeds of up to 54 Mbit/sec. Moreover, they operate over an unlicensed frequency spectrum, which makes their deployment relatively easy as no permissions by authorities in order to set-up a WLAN access point in a public place are needed. The disadvantage of operating in unlicensed spectrum is that the risk of interference from other technologies and other access points using the same frequency range is higher.

Nowadays WLAN are especially deployed in corporations or university campuses. Access to a WLAN network is usually guaranteed only to authorized users, such as the employees. In general, WLANs are mainly used in locations like houses, university departments or offices, which cannot be considered publicly accessible by anyone passing by. The utilization of WLAN to cover network access in public locations, so that a nomadic user can attach to an access point having no previous security association with it, is limited but growing. Main reason for

[9] If the High-Speed Packet Access (HSPA) technique is used, the theoretic limit is 14.4 Mbit/sec downlink and 5.76 Mbit/sec in uplink

this situation is that currently, the most widespread wireless device that supports the WLAN technology is the laptop, which is nevertheless not as ”portable” as ubiquitous communication would require.

While some PDAs are equipped with WLAN cards, they have not penetrated so much into the mass market. We believe that a real extensive deployment of WLAN in hot-spots will be possible only when hand-held devices for the mass market will be equipped with WLAN capabilities; these devices are the mobile phones. Currently, only few of the high-end segment phones have WLAN capabilities, which, however, are practically unusable, as they drain the battery too quickly. With the improving of batteries technology, these limitations will likely become less severe. Indeed, a major problem is that, unlike the 3G system, the WLAN technology has not been natively designed to be run in small battery-powered devices. Our prediction is that in the near future low-end phones with WLAN capabilities, together with 3G cellular network access, will be available for the mass market. Designing services that use the capabilities of both these access methods is crucial to ensure a wide deployment of these devices.

As already discussed, applications that could become the killers of this new ubiquitous communication paradigm, involve real-time exchange of data among users. Voice over IP (VoIP) could be one of those applications. More efficient link-layer Medium Access Control (MAC) protocols are also needed to overcome the performance problems of the current WLAN 802.11 MAC protocol. Especially when time sensitive applications, such as VoIP are run over WLAN MAC, several performance problems arise [134]. A key factor for widespread development of WLANs is therefore the realization of an efficient medium access mechanism, which is not necessarily the one currently deployed. New wireless access technologies, such as IEEE 802.15.3, may instead take over WLANS, and be deployed due to their higher data rate and better medium access technology.

Mobile devices of the future will have enhanced communication capabilities, and more processing and battery power. If mobile services and applications able to exploit these augmented terminal and network capabilities will be designed for use in handheld mobile devices, people would carry more than a simple phone, rather a small computer. The concept of Personal Area Networks (PAN) [127] will be extended from the idea of devices located in a restricted area, to devices that are always carried along.

3.9 WLAN Deployment Scenarios

This section presents deployment scenarios for WLANs, to be seen as extended ubiquitous communication networking environment. The scenarios we outline must be analyzed with the mind projected in the future, as the current technology

is not yet capable to fully support the technical requirements that such scenarios pose.

Basic assumption is the availability of a device with WLAN and cellular network access capabilities; further, in order to achieve truly ubiquitous communication computing, simultaneous access to heterogeneous networks must be supported. Devices should be able to join the network through a WLAN access-point or in ad-hoc mode, via a direct connection to other peers.

A device with such capabilities truly enables users to carry their "virtual extended PAN" in the pocket; the user will have diverse connection capabilities to fit all his needs. He may use WLAN in ad-hoc mode to exchange instant messages with nearby people; alternatively, he may connect to the WLAN access point of a mall to retrieve a list of the offers on the products he is interested in. At the same time, he may engage in a voice conversation over the 3G circuit-switched network with a friend in another city.

3.9.1 WLAN as Ad-Hoc Networks

Utilizing the services of an access point is not always needed or possible. The deployment of WLANs in ad-hoc mode is a powerful method to exploit ubiquitous communication capabilities. We do not deem WLAN as the only feasible technology for realizing ad-hoc networks; in some scenarios, other existing technologies like Bluetooth, or new solutions yet to appear, may be more suitable. Nevertheless, WLAN is a promising technology for ensuring ad-hoc communication, and we use it in this section as the example technology for supporting future ad-hoc networking scenarios.

In this chapter we have so far presented technical issues related to the deployment of ad-hoc networks in general, regardless of the underlying technology used to provide connectivity. Here we discuss example use cases of ad-hoc networks, where utilizing WLAN technology is an effective choice. The scenarios have been introduced already in Section 1.1; in the outlined vision, little was said about the technology that could enable such computing environment. WLAN-based ad-hoc networking is a good choice; it provides reasonable bandwidth capabilities and does not force devices to be too close with each other. Bluetooth, which is the technology that is most utilized nowadays for building small (usually formed by two nodes) ad-hoc networks, has capacity problems.

In the mall scenario, people can use ad-hoc (or proximity) capabilities to contact other users in the area, without involving any infrastructure node but relying the communication only through the peer nodes. The nodes forming the hops of the ad-hoc network would be the customers, who move around while they shop. Access points would only be used to retrieve services from the network, like Internet access or connecting to the banking service with the personal device and

paying for an effected purchase. Ad-hoc networks based on 802.11 technology can be set-up to enable infrastucture-less communication among the members of a rescue team in a disaster area.

One of the technical difficulties to solve is the ability to transparently and quickly set-up WLAN-based ad-hoc networks. Currently, setting-up an ad-hoc network using WLAN requires configuration efforts, which may discourage those who are not technically expert to use the technology.

3.9.2 Interoperability with Fixed Networks

The coexistence of 3G and WLAN access capabilities in the same end device makes the need for network interworking natural and compelling. An efficient interworking solution must ensure that a device can seamlessly move from one network technology to another, without suffering connectivity disruptions. Interworking must be guaranteed at service level, and service continuity must be guaranteed also after a vertical inter-technology handover. For example, a VoIP session should not undergo QoS disruptions during a handover. This would ensure a better user experience and increase the probability that ad-hoc communication could grow as popular as cellular communication.

The concept of integration of heterogeneous network can also be extended to the possibility of having connectivity to both network types at the same time. For example, a user may be involved in a chat session with two friends. One friend is in a neighboring area, and connectivity between devices is achieved through WLAN in ad-hoc mode. The other friend is instead reached through the 3G network. Fundamental requirement for the technical realization of this scenario is that devices are multi-homed, and can support more than one IP address at a time.

An architecture for integrating WLAN ad-hoc networks with cellular networks is presented in Section 6.1 of the book [127]. The basic idea motivating the architecture is extending the capabilities of a cellular network by creating smaller WLAN-based cells, where higher data throughput can be guaranteed. The goal is enabling ubiquitous communication with high resource availability. The architecture is referred to as *Hierarchical Multihop Cellular Network*.

The hierarchy in the multi-hop cellular network is formed by the size of the cells. The cellular network cell is the highest layer of the hierarchy. It is used by the mobile terminal as a fallback solution when connectivity with the lower level cells is not available. Communication is carried out over the IP network, and the base station serves as relay to the backbone packet-switched network. Directly connected to the IP backbone are WLAN access points (AP), forming a second-level hierarchy of cells. Several multi-hop capable nodes operate in the coverage range of the APs. A multi-hop capable node is in charge of a subcell (the third level of the hierarchy); the mobile terminals in the coverage area of the subcell

communicate in ad-hoc mode with the multi-hop capable node.

With this structure, higher bandwidth values than those provided by pure cellular networks can be achieved. Moreover, the higher layer entities like base stations or access points, can provide the terminals in the ad-hoc subcells with useful information related, for example, to routing. Additionally, they can enforce effective resource management and load balancing policies in the lower layer cells.

Another possibility for integrating a WLAN-based ad-hoc network with the infrastructure, is using the WLAN technology itself. A node in the ad-hoc network can discover the presence of a gateway to an infrastructured network, like the Internet. The gateway node can be another peer forming the ad-hoc network, or a fixed node that has an interface to the infrastructure. The fixed node does not act as an access point, but it is contacted by the other nodes using WLAN ad-hoc mode connectivity.

The emerging wireless technologies open new scenarios for the interoperability of ad-hoc and proximity networks with infrastructured networks. Access to the IP backbone can be obtained, e.g., through WiMax, while connectivity in the ad-hoc domain through IEEE 802.15.3. The ideal situation would be to obtain seamless interaction between any kind of ad-hoc and proximity technology (WLAN, 802.15.3) and any kind of wireless access networks (3G, WiMax, WLAN through an access point).

Technical issues related to this approach lie, e.g., in the mutual authentication between the gateway and the ad-hoc nodes. If the gateway is a fixed node, it may be provided by an operator; it can therefore provide its credentials and verify those of a user trying to access its services. Acceptance of the gateway credentials by the ad-hoc node is in this case subject to some degree of trust. In fact, the only way for the ad-hoc node to verify the gateway credentials is, unless they have been previously locally stored, to access a verification service in the Internet. If access to the Internet is obtained through the very same node whose identity is to be verified, we can see how it is not possible to guarantee in all the cases full security for a node in the ad-hoc network, as it would be possible that a malicious gateway node could forge the response sent to the ad-hoc node.

Chapter 4

Message Spreading Techniques

The client-server paradigm has different characteristics compared to the distributed peer-to-peer paradigm. One difference is that the service is provided by a well-known entity, rather than by the network in its whole. This characteristic affects directly the type of communication in client-server systems: usually clients connect to the well-known server address and engage in unicast communication with it. This assumption is not generally true in distributed systems, where it is necessary to find the address of users and services in the network before being able to reach the node hosting that service. Such a search for services is often done spreading a message, containing, for example, the query data, into the network. The query should efficiently reach a big enough number of nodes and a reply to the query should always be received, if the item is found. An alternative is for a node to communicate[1] its own information about hosted services or contact data to all the other nodes in advance, to advertise in the network its presence and the services it provides.

In order to have efficient query propagation, message spreading should ideally continue until a reply for the query is found, or a notice that the query cannot be answered is returned to the querying node. The querying process should not continue after that point, in order to avoid sending unnecessarily messages to the network, which is an issue especially for low-bandwidth environments, such as some wireless networks.

The unicast paradigm clearly does not fit this model of spreading information. Contacting a single node is often not sufficient to retrieve information about users and services in the network. Neither is it sufficient nor always efficient to ensure that information about own provided services is made available to a large number of users. Other methods are needed.

This chapter addresses various methods for spreading information in environ-

[1] Such an operation is referred to as publishing in some systems

ments where the support of a centralized entity is not available, such as ad-hoc networks. We show in which scenarios a technique is good, pinpointing its advantages and disadvantages. Pointers to related literature are provided as well. The discussion does not aim at being exhaustive on all the possible methods for disseminating information in ad-hoc networks, but at giving an overview of the main techniques. There are two main motivations for message dissemination to a large number of nodes in an ad-hoc network. The first is related to routing, to discover the path to destination. Spreading information can also be used to distribute data among nodes in the network; user data distribution to a large set of nodes is a peculiar characteristic of peer-to-peer systems, which are not necessarily deployed in ad-hoc networks. However, the decentralized nature of the ad-hoc environment makes efficient distribution of information an active research topic.

4.1 Broadcasting

Broadcasting is the simplest way of disseminating information in decentralized environments. Broadcasting means that a node sends a message to all its link layer neighbors. We underline that for technologies where the medium access is shared, like Ethernet or WLAN, sending a unicast message to a specific IP address means broadcasting it at link layer. The message will reach all the nodes sharing the medium. The difference with the broadcasting case is that unicast messages addressed to a different user than the local one, are filtered out by the network cards connected to the medium. Broadcast messages are meant for all the users in the subnetwork, and are therefore processed by all the nodes and not filtered by the network cards. Broadcast messages are generally (exceptions may depend from local policies) restricted to the subnetwork where they are sent. Sending broadcast messages in ad-hoc networks implies that they reach all the one-hop neighbors of the sending node.

Broadcasting messages is a simple alternative in small ad-hoc networks, especially if the nodes are all connected with each other at link-layer. There is no state information to keep, no routing tables to maintain for nodes in order to use broadcasting. In IP networks it is necessary simply to properly set the destination IP address to the broadcast address and send the message to the network. The major drawback of broadcasting, which is a common disadvantage for all the techniques for spreading information to large number of nodes, lies in its scalability. If the network grows bigger, and nodes keep on sending messages, the network could become congested. A considerable amount of research has been done on broadcasting in ad-hoc networks. In practically all the cases, broadcasting is used as a synonym of flooding and applied to multi-hop networks. We make a difference between the two terms, referring to broadcasting when talking of link-local ad-hoc

networks and flooding when extending the technique to multi-hop networks.

In summary, broadcasting is a simple technique, which can be used in small networks or in situations where the rate of broadcast messages is low. In such scenarios, the simplicity of the approach is preferred to more complex algorithms, like those discussed in Section 4.3, aiming at saving bandwidth and reducing the number of messages sent. Such savings would in fact come at the expense of higher processing burden on the nodes and possibly of keeping algorithm related state information.

4.2 Multicasting

Multicasting a message means sending it to a subset of nodes in the network. From a logical point of view, broadcasting is a particular case of multicasting, where the subset of nodes is comprised of all the nodes in the network. Another relevant difference between multicasting and broadcasting is that in most of the cases, multicast messages traverse routers. The way of using multicast messages is to register to a multicast group; a multicast group is assigned a particular IP address (in IPv4 multicast addresses belong to the class D of addresses). All the nodes registered to that group receive the messages sent to the multicast group IP address.

Multicasting has the advantage that messages reach only the group of hosts that are the intended recipients. Hosts that are uninterested in the message, or that are not supposed to receive the message, will not receive the message, contrary to the broadcast case. This feature comes at the expense of defining algorithms for creating and maintaining multicast groups and multicast routing protocols. The aim of these protocols is to make sure that the multicast message is distributed to all the intended recipients, the least possible number of routers in the path is involved, and that, when no longer needed, the routes created for multicast purposes are removed.

In ad-hoc networks, the efficiency of multicast routing is even more important than in wired networks, for the capabilities limitations of ad-hoc nodes. Several algorithms for multicast routing in ad-hoc networks have been presented. Royer and Perkins have proposed multicast extensions to the AODV protocol (MAODV) [108]. Multicast routing is achieved by building a distributed and on-demand multicast tree. Nodes that are members of a multicast tree, i.e., they have routing responsibilities for a particular multicast address, need to keep a multicast routing table besides the unicast one. A different routing approach is chosen for the On-Demand Multicast Routing Protocol (ODMRP) [65]. ODMRP does not build multicast trees but uses a mesh-based technique leveraging on the concept of a forwarding group, where only a subset of nodes forwards the multicast packets

via scoped flooding. A comparison of the two protocols is presented by Kunz and Cheng [63]; the result of their simulation study is that ODMRP outperformed multicast AODV due to the availability of alternative paths, which allows multicast datagrams to be delivered to all or most multicast receivers even if links fail. There are also other techniques for multicasting; two approaches based on probabilistic methods are discussed later in Section 4.4.

Multicasting is preferred to broadcasting in multi-hop ad-hoc networks, where broadcasting cannot be applied to nodes that are more than one hop away. Another advantage of multicasting is that only nodes that are interested or are supposed to receive the messages will receive them. The operations of building and maintaining multicast routes pose a burden on ad-hoc nodes; therefore, for small link-local ad-hoc networks the simplicity of broadcasting is still preferred.

4.3 Flooding

Flooding is an extension to broadcasting allowing messages to be forwarded also over one-hop neighbors. While the nature of a message, whether it is broadcast or multicast, can be understood simply by analyzing its IP header, flooding is triggered by an application or by the routing algorithm.

In the most basic form of flooding, the nodes that receive a flooded message, broadcast it through all their network interfaces, except the one from which the message was received. In ad-hoc networks, enforcing this rule is easy, as nodes have usually only a single interface connected to the network. The basic algorithm is clearly inefficient, as it does not provide means to stop message forwarding. The most used technique to interrupt the flooding is to use a hop counter and have nodes decrease it by one before forwarding a message. When the counter reaches zero, the message is no longer forwarded. The initial value of the hop counter usually depends on the network topology or on the estimated distance of the recipient node from the source. Another improvement consists of keeping track of the packets that have been already forwarded, to avoid forwarding them again when loops are generated during the forwarding process.

Efficient flooding in ad-hoc networks is a very sensitive area of research, several proposals for improving the basic flooding techniques have been presented. Sheng et al. [117] characterize the problems that flooding poses in ad-hoc networks into three categories:

- Redundant broadcasting: nodes broadcast messages to neighbors that already possess the message
- Link-layer contention: neighboring nodes attempt to transmit the broadcast message at the same time

- Collisions at link-layer: as direct consequence of contention

The overall effect of these problems is referred to as *broadcast storm*. Ni et al. [87] provide a simulation based study on the performance of various techniques for improving the basic flooding algorithm and alleviating the broadcast storm problem. The common denominator of all the proposed techniques is to prohibit some nodes from rebroadcasting a message to reduce the occurrences of redundancy, contentions and collisions. For example, in probabilistic flooding, nodes rebroadcast a message with a certain probability P. Probabilistic flooding coincide with basic flooding for P = 1. Tuning the value of the parameter P has been a popular subject of research; several papers propose different algorithm for determining the probability of flooding.

Applying flooding in ad-hoc networks is often a forced choice, since flooding, despite the overhead it causes, is the approach that best fits networks with dynamic topologies. E.g., the AODV protocol [90] uses flooding for performing route discovery. The techniques proposed to improve the basic algorithm do not provide a definitive solution, but they are rather optimized for particular types of ad-hoc networks. For example, some techniques designed for ad-hoc networks with highly dynamic nodes may not perform well in more static environments and vice versa. The target size of the network is also a major influence factor.

A summary paper on flooding (in the paper referred to also as broadcasting) algorithms for ad-hoc networks is provided by Williams and Camp [137], who analyze and compare different approaches. The protocols are evaluated in a range of network environments comprising node density, node mobility and traffic rates. References to all the discussed protocols are provided, together with a categorization of the algorithms. The interested reader can refer to the paper for a first overview on flooding algorithms, and use the provided references for more details on the single algorithms.

4.4 Probabilistic Dissemination Algorithms

Probabilistic dissemination algorithms belong to the family of probabilistic flooding algorithms. They all have in common the fact that a received flooding message is forwarded with a certain probability < 1. There are several variants, e.g., in some algorithms, randomization is applied not only to decide whether to forward or not, but also to select a subset of neighbors to which to forward the message. Epidemic algorithms are an example of probabilistic dissemination algorithms.

Research on epidemic algorithms was ongoing already in 1987 [24], even though then it was applied to increase consistency when spreading information in distributed databases. The mathematical theories behind epidemic distribution of information are even older; a book on the topic is dated 1975 [5]. The basic

idea behind epidemic algorithms is that information should be spread in the same way as diseases, where individuals who randomly get in touch are infected. In distributed computing systems, this means sending a piece of information (the theory is general, applied to networking the piece of information becomes a message) to a randomly chosen set of peers.

There are three parameters affecting an epidemic algorithm [29]. The first is the buffering capacity of a node, in terms of messages that it is possible to store before they are forwarded. The second parameter regulates the number of times a message can be forwarded. The third parameter affects the number of nodes to which a message can be forwarded. The various epidemic algorithms differ in the way these parameters are utilized, i.e., whether they are fixed or dynamic, or whether they are affected by network topology and so on. A fundamental property of epidemic algorithms is that they express the so-called *bimodal behavior* [29]: messages are all or not at all delivered with very high probability, while they are partially delivered with very low probability.

Epidemic algorithms are often referred to as gossiping algorithms. Active fields of application in ad-hoc networking are routing and distribution of cached information among ad-hoc nodes. Haas et al. propose a gossip-based routing algorithm [42]; they observe that gossiping presents the bimodal behavior and find out that it is particularly affected by the gossiping probability and by the network topology. They find that a gossiping probability between 0.6 and 0.8 suffice for ensuring high reachability of nodes in the network and allows reducing the number of messages sent to the network by 35% compared to basic flooding.

A variant of gossiping is anonymous gossip [17]. Anonymous gossip addresses the issue of reliability of multicasting in ad-hoc networks. The goal of the algorithm is to improve packet delivery of multicast routing protocols and decrease the variation in the number of packets received by different nodes. The protocol is not itself used to route multicast messages to a group, but to recover those messages that got lost during the routing process from the members of a multicast group that might have received it. The method is said anonymous because nodes do not need to know the other members of the multicast group in order for the gossip to succeed. Bimodal multicast requires instead an at least partial knowledge of the multicast group members. Anonymity is a desirable property in ad-hoc networks, as it leverages ad-hoc nodes, which often have limited resources, from having to maintain group membership information. Gossiping is anonymous as messages are forwarded to randomly chosen neighbors, regardless of whether they belong or not to the multicast group. The actions triggered at the node that receives a message depend on whether or not the node belongs to the multicast group.

Route driven gossiping [73] is another variation of the basic algorithm, also

aiming at providing reliable multicasting in ad-hoc networks. The claim of the authors is that a probabilistic approach ensures more scalability than a deterministic one, when targeting reliability of ad-hoc multicasting; this consideration motivates their choice of a probabilistic algorithm for reliable multicast. Route driven gossiping tries to overcome a major pitfall of anonymous gossiping, which relies on the Multicast AODV protocol [108] for spreading multicast messages, and limits de facto its scope to the scope of MAODV. Another consequence of this dependence to MAODV is that anonymous gossiping loses one of the properties that make gossiping algorithms attractive, i.e., the predictability of behavior, in terms of nodes that will receive the information. The route driven gossip aims at reliable and predictable multicast routing.

The protocols discussed so far are essentially targeted at improving basic ad-hoc routing techniques. Autonomous gossiping [23] instead aims at finding an efficient way to spread and replicate cached information in the nodes of an ad-hoc network. In autonomous gossiping, the data items themselves try to identify other hosts which may be interested to the item, based on the data item's own profile and host's profile, advertised during registration phase. This approach is in contrast to the traditional push model were data items are injected in ad-hoc networks by the possessor nodes. Profiles are maintained in a distributed self-organizing way, and updated using gossiping techniques. When data items arrive at a node, the autonomous gossiping algorithm is applied to decide what the data item will itself do, in an autonomous and self-organizing fashion. Data items in a host decide whether continue to reside, migrate or replicate to another host.

Another interesting approach is described in [37]. The authors propose to enhance data availability in sensor networks by having sensors randomly distribute their cached contents to a random set of neighbors. If data received with a message do not fit into the small sensor node cache, then existing data are replaced by the new data. This shuffling ensures that an immediate neighbor gets a replica of the information being spread, and at the same time allows redistributing evenly the data among all the nodes of the network. Despite its applicability scope is definitely out of target for our chosen network environment, as it addresses sensor networks with possibly thousands of nodes, it is an interesting option for spreading user data in ad-hoc networks.

4.5 Distributed Hash Tables

Distributed hash tables (DHT) take a completely different approach for disseminating information. DHT algorithms have been traditionally applied to search operations in peer-to-peer systems running with the support of a large infrastructured network in mind, e.g. the Internet. The general idea shared by DHT algo-

rithms is that nodes and data items are assigned an ID; when searching a data item with a particular ID, the system routes the query message towards the node with the closest possible node ID, with respect to the queried data ID. Responsibility for a given data item is assigned to a given node based on a hash function, like SHA-1. The core of the DHT algorithm is to provide a look-up service for allowing a node to quickly and efficiently discover the node that is managing the data item.

Nodes in a DHT are organized according to an overlay network, which assumes a different topology depending on the algorithm (i.e. circle or a hypercube). The logical identifier of each node determines its logical position in the overlay. The logical identifier is acquired by a node during bootstrap phase, or it is known in advance. Bootstrapping into the existing system allows joining the overlay DHT network. Usually join operations are done by contacting a node that is known to be in the system already, but variations are possible.

Bootstrapping is a very delicate phase in DHT overlay networks; it may require lots of operations and message exchanges, and be time consuming. When the join procedure is completed, the newcomer gets to know a set of (logical) neighbors and other information necessary to build the node's routing table. The bootstrapping overhead constitutes a major difficulty for deploying DHT in ad-hoc networks. Another limitation of DHT is that while two nodes may be neighbors in the overlay network, they can be physically located very far away from each other. Some DHT algorithms try to improve the efficiency of DHT neighbor discovery taking into account also geographical vicinity, rather than the pure outcome of the hashing. Routing tables are used by DHT nodes to efficiently determine what other node is responsible for a given piece of data.

These operations are practically common to any DHT algorithm; the differences lie in how the logical partition of the overlay network is done, how the data ID are assigned, how the routing tables are maintained and so on. Examples of well-known DHT algorithm are Chord [123], where the nodes are organized in a logical overlay ring, and CAN [97]. CAN targets the scalability of the indexing mechanism in a virtual d-dimensional Cartesian coordinate space on a d-torus. The physical network that is thought to be under the overlay network, in both cases, is a large scale network, like the Internet.

There is not much work done on adapting DHT algorithms for use in ad-hoc networks. A proposal for efficient adaptation is given by Pucha et al. [94], where bandwidth limitations, node mobility, and multi access interference are identified as the challenges to face for deploying DHTs in MANETs. The authors first propose to directly overlay a DHT on top of an ad-hoc routing protocol, then the alternative way to embed DHT operations at network layer using a newly defined protocol.

4.5.1 Case Study: P2P SIP

An individual submission to the SIPPING WG [13], called P2P SIP (peer-to-peer SIP), discusses how DHTs can be applied to SIP registration procedures to make them completely decentralized. The proposal uses the peer-to-peer overlay network constituted by Chord [123] to spread user's registration information over multiple nodes.

Although SIP can be natively used as a peer-to-peer protocol, centralized servers are in most of the cases needed to manage contact information of users in the network and for reachability reasons. Thus, SIP has always been considered in connection with servers. This proposal to make SIP usable also without servers has quickly gained noticeable interest in the research world, and implementations and related work are numerous. In IETF, besides the main specification ID [13], several other IDs are being published dealing with related arguments, which is even more relevant if it is considered that the P2P SIP draft is still an individual submission. To proof the growing interest towards P2P SIP, recently a new dedicated working group has been chartered in IETF[2] to discuss the implications of deploying SIP without support of centralized servers. At the time of writing this dissertation, the P2PSIP WG has not yet produced its first draft.

Analogously to Chord, P2P SIP nodes build an overlay network on top of the SIP network; nodes joining the network must first register to the overlay network and only after that they can perform, in a decentralized fashion, the traditional operations of SIP user registration. This involves communicating the mapping between their user name and IP address. The system is a conventional DHT based network: each node has an identifier, and each SIP user name (AOR) is mapped (hashed) into a resource identifier. Nodes in the overlay network will manage resources whose ID is comprised in the hash region space of their competence, that is, is close enough to the node ID of the handling node. Close enough depends on the conditions of the overlay network, in terms of number of nodes, their ID, and ID of the resources currently in the network, that is, which users are on-line.

In any case, the hash space is divided such that a resource will always be assigned to at least one node's portion of space. Each resource is redundantly stored in more than a single node, to provide robustness against node failures or voluntary departures. Each node in a P2P SIP network can act as a user agent, a registrar or a proxy server, and maintains a fraction of the information on the currently on-line users in the network. Nodes have knowledge on how to contact a number of other nodes in the overlay, called the neighbors of the node. Due to the properties of the DHT algorithm used, usually neighbors will have a node Id similar with each other.

[2]Peer-to-Peer Session Initiation Protocol (P2PSIP) IETF working group: `http://www.ietf.org/html.charters/p2psip-charter.html`, accessed October 9, 2007

When a node tries to locate a resource with a given ID, it looks in its neighbor table and addresses the query to the node with the node ID closer to the resource ID requested. If the contacted node does not handle the resource, it will forward the request to the neighbor with yet a closer node ID. The process is repeated until the resource is located. These operations are normal in DHT algorithms; what is particular in P2P SIP is that all the messages exchanges are performed by means of SIP messages.

In order to register to the overlay network, a node must first contact a bootstrap node. The address of the bootstrap node can be either preconfigured, well-known, fetched with out-of-band mechanism or after broadcast query. After that the bootstrap node has been located, the joining node sends to it a SIP REGISTER message. The To and From fields of the message contain the hashed ID, the user name and the IP address of the registering node. A new defined header field, DHT-NodeID, contains the Node ID and IP address of the registering node. The bootstrap node examines the Node-ID to determine if it corresponds to the portion of the overlay the bootstrap node is responsible for. If not, the bootstrap node will return to the registering node a SIP 302 Redirect response containing the ID of the node in its neighbor table nearest to the registering node ID. The received address will be used by the joining node as the new bootstrap node. The process is repeated until the node contacted is currently responsible for the area of the DHT in which the new node will reside.

The receiving node that is responsible for that portion of the overlay is referred to as the admitting node. The admitting node sends to the new node a SIP 200 OK message, containing in a new defined header field, DHT-Link, the contact information of admitting node's predecessor, successor and 4 next entries in its neighbor table (for more details on the operations see [13]). These addresses will be used by the joining node to perform redundant registrations. We do not go into further details of maintaining the overlay network, and updating it after events such as join or leave. When a node is in the overlay, it must register the user(s) for which it is responsible into the overlay as data. User registration adds into the overlay a binding, treated as resource in the overlay network. After the node has hashed the user's AOR, it looks in its constructed neighbor table for the node whose ID is closer to the computed user's resource ID. The REGISTER is forwarded and then treated exactly as a node registration in the network.

DHT deployment in ad-hoc networks is a challenging task, due to bandwidth limitations, node mobility, and multi access interference [94]. Especially if the ad-hoc network is small or even link-local, the burden of maintainance a DHT table can be too much compared to other simpler approaches. Indeed, the native target network environment of DHT is the Internet. In proximity networks, where all the nodes can easily communicate with each other, it makes little sense to build

an overlay network. In P2PSIP case, the amount of signaling messages needed to register to the DHT overlay network and to maintain DHT routing tables can result in an excessive overhead for ad-hoc deployment. Redundancy of registrations increase reliability at the expense of signaling overhead and bandwidth consumption.

4.5.2 Related Work to P2P SIP

This section briefly describes few IDs and other research work related to the main P2P SIP draft. In the draft [54] the authors appreciate the efforts for developing SIP over a P2P network, but criticize the fact that SIP messages are used for registering to the overlay network. They claim that the approach proposed in P2P SIP [13] may increase the load on the network and search delays. A technical report from Columbia University [119] also addresses SIP deployment in a P2P fashion. The report lists the advantages of a P2P approach over the traditional one, most important of all being the scalability of the system and absence of maintenance costs, but also the disadvantages in terms of higher lookup failures or response time, and relevant security threats.

Baset et al. [8] list the requirements that P2P-based SIP Internet telephony applications should have. High-level, architectural and protocol requirements are given, as well as some example use cases of P2P SIP. Note that the draft does not explicitly refer to the P2P SIP proposal discussed above [13], but rather to a generic SIP P2P architecture, thus these requirements also apply to decentralized SIP, proposed in this dissertation. One of the target application environments for P2P SIP services is small-scaled ad-hoc networks, while a major requirement is small signaling overhead. Despite being general, the requirements imply the use of DHT for locating and managing SIP users information. While DHT may be the best choice in world-scaled networks, like the Internet, their usage in small-scaled ad-hoc networks is certainly an overkill, due to the high overhead that DHT tables maintainance causes. A more complete set of use cases for P2P SIP can be found in the draft [14].

An architecture for a P2P SIP network is illustrated by Shim et al. [118]; basically, the document describes a generic architecture for building P2P-based SIP systems. It defines the logical entities forming the P2P architecture, how bootstrap operations can be performed, and an API for the DHT overlay network built on top of SIP and security considerations. P2P SIP [13] can be considered as a concretization of the generic architecture defined in the draft [118].

The most interesting aspect, and challenging technical issue, related to P2P SIP (and, in general, to any P2P system), is solving the NAT traversal problem. The purpose of SIP servers in session establishment phase is not only of resolving a SIP AOR into its contact address, but also to help the signaling in case one

or both UAs are behind a NAT. If all the signaling is peer-to-peer, the issue of NAT traversal for signaling messages need to be addressed. The requirements presented by Baset et al. [8] suggest to distribute NAT traversal capabilities in the network among all the peers, which is probably the most feasible solution, and it is already utilized with success in currently deployed P2P commercial Internet telephony services, like Skype.

Part II

A Decentralized Approach for SIP

Chapter 5

Decentralized SIP

The second part of the dissertation presents and evaluates in detail our solution for decentralized session management. We describe its features and functionalities, present implementation details and discuss the security support with which it has been enhanced. We also provide an analysis of the signaling overhead of our solution, with and without security support. We describe how the solution can be deployed in link-local ad-hoc networks, and the extensions needed to adapt it to a multi-hop network environment. We refer to our decentralized session management solution as *decentralized SIP*.

5.1 An Introduction to Decentralized SIP

This chapter analyzes the design of decentralized SIP, of which two versions have been developed. Those versions differ in the way registration operations, which in ad-hoc networks also involve discovering which users are available, are performed. The first version, called *fully distributed SIP or dSIP*, uses SIP-only messages to register a node in the ad-hoc network. The second version, *SLP-aided SIP or sSIP*, uses instead the SLP protocol in the registration phase. These two versions of decentralized SIP are optimized for use in small ad-hoc networks, comprised of few nodes.

To complete the design of decentralized SIP, we present SIPCache, an extension to the basic decentralized SIP functionalities allowing to efficiently deploy decentralized SIP in bigger multi-hop ad-hoc networks, with several dozens, or even hundreds, of nodes. SIPCache leverages on PCache, an algorithm used for efficently distributing and caching data items in ad-hoc networks, by means of replicating data items using geographical distance from the sender, and some elements of gossiping. We underline that decentralized SIP does not need SIPCache in order to be deployed in multi-hop ad-hoc networks, but it can run transparently

on top of any ad-hoc routing protocol, or it can rely on multicasting of messages in the ad-hoc network.

Besides the middleware layer constituted by SIP, a so called proximity manager application was developed for executing decentralized SIP signaling exchanges; this application is a session manager, and is used to manage decentralized SIP sessions. This application allows, e.g. registering a user's SIP contact information within or outside ad-hoc networks, beginning and managing SIP sessions. Particularly, we used the proximity manager application to establish instant messaging sessions, based on the Message Session Relay Protocol (MSRP) [26]. We have implemented our own MSRP library and realized MSRP instant message exchanges in ad-hoc networks, using decentralized SIP as the signaling protocol. MSRP instant messaging constitutes an example application that can utilize the distribute signaling capabilities of decentralized SIP.

The session management framework in ad-hoc networks has been enhanced with security support. Using native SIP methods, modified for use in a decentralized environment like ad-hoc networks, we let users who share a previous security association with each other (e.g., they have exchanged offline their security certificate-key pair), to be guaranteed of the identity of the remote communication party, despite no centralized authorities are available in ad-hoc networks to act as third-party intermediary. This basic model has been enhanced by allowing also previously unknown users to secure their signaling exchange. In this case, once a user retrieves the security certificate of a remote party, he can be sure that for the rest of the session, and for all future sessions with that given user, the communication really happens with the user identified by that certificate. It is also possible to prompt for authorized users (those whose certificate trustfulness has been verified) to retrieve the certificates of third party users. In this case, the authorized user acts as the tursted third party entity in the security exchange. The described security architecture only applies to signaling exchanges. We leave to the signaled application to enforce the requested security level, once a session has been established.

The most interesting aspect of the framework is its full interoperability with the Internet. A real time communication framework for ad-hoc networks is in fact of little utility if it is not possible to seamlessly reutilize it in the Internet. Our framework reaches this goal, for two reasons. First, it is possible to reuse the same middleware software in an ad-hoc network, or connected to the Internet, since the middleware layer is adaptive, recognizing in which environment it is currently running, and triggering ad-hoc or standard operations as needed. Second, provided that in the ad-hoc network there is a node offering connectivity to the Internet, it is possible to establish communication sessions between a node inside and another outside the ad-hoc network. For the purpose, we have extended the

basic signaling framework for allowing this interoperability; the core of the design consists of running a SIP proxy/registrar server on the gateway node to the infrastructured network, which allows communication when the ad-hoc addressing space is not public.

To prove the effectiveness of the design, we have implemented the gateway and run testbed experiments showing how a signaling session could be established between heterogeneous networks. To complete our experimental work, we have realized, an MSRP relay node, to allow instant messaging sessions between ad-hoc and infrastructured networks. A complete and functional application was thus shown. Clearly, the Internet nodes do not need to be aware that the remote party is in an ad-hoc network, as the fact is hidden by the design of the interoperability solution. This approach is valid not only in an ad-hoc network, but can be reused to allow NAT traversal when decentralized SIP is deployed in a private network: it is enough to have the SIP proxy node running on the node acting as NAT and with an external public interface. To complete the design, we offer the possibility of secure access to the gateway node. Security support has two purposes, one to ensure ad-hoc node and gateway of mutual identity, and the other, if needed, to perform authorization of access to the gateway services. Communication between an ad-hoc node and the gateway has been secured using IPSec.

5.2 Problem Statement

The architecture of SIP is based on centralized entities. As we have seen in Section 2.2, two logical elements play key roles in the architecture, registrar and proxy servers. Registrars are the SIP entities where SIP users register their contact information once they connect to the SIP network. In the basic registration scenario, described in Section 2.5.1, a SIP user agent communicates to its registrar server the SIP user name of the user(s) using the device[1] and the addresses where the user is reachable. SIP registrars provide an abstract service, called location service, and return the bindings for the AORs falling under their domain of competence to the SIP entities issuing a binding retrieval request, usually SIP proxy servers.

Proxy servers are needed because SIP users cannot know, in the majority of the cases, the current complete contact information of the callee but only his AOR. SIP presupposes that the AOR of the party to contact is known in advance, analogously to what happens when sending instant messages or e-mails. A basic SIP session involves the calling user agent contacting the calling side proxy server, which in turn will forward the message to the proxy server responsible for the domain of the called user agent. The called side proxy server retrieves from the

[1] We recall that in SIP, user names are referred to as address of records (AOR) for a user

called side registrar (i.e. uses the location service) the bindings for the called user and eventually delivers the request to the intended recipient.

The described architecture is clearly not applicable to ad-hoc networks, as they are dynamic networks formed by peer nodes while proxies and registrars are fixed, static and centralized entities. As a result, the SIP protocol as it is cannot be deployed in isolated ad-hoc networks. SIP users in ad-hoc networks cannot reach other parties, as they do not have support from proxy servers, and cannot be reached by other nodes, as there are no SIP registrars where they can register their contact information. Decentralized SIP aims at enabling the use of SIP in ad-hoc networks, and in general, in any network environment where no support from centralized entities is available, needed or desired. We achieve distributed and decentralized session management by embedding SIP proxy and registrar server functionalities in each end node. Thus, a decentralized SIP-enabled device in ad-hoc networks runs a user agent and a SIP server. When the device is used in infrastructured networks, only user agent functionalities are utilized. The SIP server acts as a co-located preconfigured registrar/proxy server for the user agent residing in the same device, allowing each end node to reach and be reached by other nodes.

5.3 Scope of Decentralized SIP

The scope of decentralized SIP was set by identifying deployment use cases, defined in order to find suitable real life applications that could benefit from the use of decentralized SIP in ad-hoc networks. An example of ad-hoc networking scenario is a meeting; the speaker presents some slides, sharing them with the other participants on their laptops. All the laptops are connected in ad-hoc mode; while the speaker talks, all the others can see the presentation slides in their laptops. The necessary signaling for exchanging the data can be performed with SIP adapted for ad-hoc networks. The meeting secretary is typing the minutes, and the other participants can read them while they are typed. The same scenario can be obtained using the assistance of an access point; however, if some of the participants are not authorized to use the access point, slide sharing would fail. The session management operations that the use cases presented in Section 1.1 imply, can all be addressed by decentralized SIP. For example, the necessary signaling for establishing a chess game session between two users waiting for a flight in the airport can be done with decentralized SIP.

The environment that we believe constitutes the most realistic deployment scenario for decentralized SIP, based on the envisaged use cases, is small ad-hoc networks. These networks have a couple of dozens of nodes at most, possibly, but not necessarily, directly connected with each other at link layer. We refer to

an environment of small link-local ad-hoc networks also as proximity networks (see Chapter 3). Ad-hoc networking research has often focused on large scale ad-hoc networks, with several hundreds of nodes. While targeting large ad-hoc networks may be more general, the practicality of such big networks should be taken into consideration. To achieve large scale deployment of ad-hoc networks in the future, we claim that realistic scenarios, possibly use case driven, should be addressed in research. We believe that large ad-hoc networks do not constitute a realistic use case for session-based real time applications.

A similar vision about realistic sizes of ad-hoc networks is given by Tschudin et al. [129]. We have already discussed their point of view in Section 3.4, but we recall it here as it coincides with ours. They affirm that simulation-based studies of large networks do not constitute a realistic ad-hoc networking scenario. They envisage ad-hoc networks with 20 nodes at most, only a couple of hops away from each other. We stress that, when possible, direct connection at link layer among all the ad-hoc nodes should be realized and preferred over multi-hop communication. Link-local connectivity would leverage nodes from running resource consuming ad-hoc routing protocols, making the overall operations in ad-hoc networks simpler.

Nevertheless, to prove the effectiveness of decentralized SIP from a more theorethic point of view, we have designed it to be efficiently deployed in big multi-hop ad-hoc networks, with hundreds of nodes. To conclude this chapter, we show how decentralized SIP can be used in bigger ad-hoc networks, introducing an algorithm that combines routing of messages and cooperative data caching to enable the SIP location service in such an environment. As also the simulation results show, by combining decentralized SIP with PCache, an algorithm invented to allow efficient distribution and retrieval of data items in ad-hoc networks, it is possible to track the bindings of users also in very big ad-hoc networks.

We recall again that decentralized SIP can be deployed on top of any ad-hoc routing protocol; for example, decentralized SIP messages can be spread in the network using multicast instead of PCache spreading algorithm. Independent research [60, 78] has shown that the decentralized SIP can be run also on top of the AODV routing protocol [90], which confirms our claim on decentralized SIP independence from underlying routing protocols.

Decentralized SIP is not limited to ad-hoc networks only, but it can also be applied to fixed networks, and in general to all networking environments where a distributed paradigm is desirable. For example, decentralized IP may be implemented in the desktop machines in a university campus and run over an Ethernet LAN. Due to the wired nature of the communication medium, in this case a larger network, like the campus is, can be taken into consideration as operational environment for decentralized SIP.

5.4 Design Goals

The design of decentralized SIP was driven by requirements, posed by the particular nature of the primary operational environment. The overall target was to build a comprehensive signaling solution that can support standard and ad-hoc SIP operations, transparently for the end user and in a secure fashion.

Our first design choice was to make all the ad-hoc nodes peers with respect to the operations of decentralized SIP. We evaluated other alternatives, such as the possibility of electing some nodes as cluster heads so that they act as registrar and proxy servers for all the nodes in their cluster. A similar proposal for the use of SIP in ad-hoc networks was also presented in [7]. We discarded this possibility because in our main target network environment, small sized ad-hoc networks, the overhead due to electing a cluster head and replacing it in case of its failure or voluntary leave, was deemed too high.

Another design goal was compatibility with the standard SIP protocol. Decentralized SIP does not substitute the SIP stack of a node, but rather complements and extends it. From the design point of view, implementing this requirement means that no new messages or header extensions to the standard protocol must be defined (unlike another proposal for adapting SIP in ad-hoc networks, where a new header field was defined [61]). Rather, we designed decentralized SIP so that ad-hoc operations are enabled by the semantic of the messages, such as the particular format of header field values, and by the addition of new software components to mobile devices used in ad-hoc networks.

Transparency to the end-user was another design goal. The human user should not use differently his SIP applications, based on whether he is in ad-hoc or infrastructured networks. Certainly, he can be made aware of whether the device is operating in ad-hoc mode or not. In any case, all the operations needed to operate SIP-based applications in ad-hoc networks are hidden by the lower layers of the decentralized SIP software architecture. With this arrangement, it is possible to use decentralized SIP in heterogenous environments, to communicate within ad-hoc networks, between ad-hoc networks and infrastructured networks and in infrastructured networks only[2]. By embedding the operating networking environment decision logic in the lower layers, it is also possible to deploy third party SIP applications in ad-hoc networks, without modifications to the original application. According to this logic, we did not develop our own entire SIP stack, but we modified an existing solution to fit the requirements of ad-hoc networks (see Chapter 6).

The last design goal takes into account the main decentralized SIP operational environment, i.e., wireless ad-hoc networks, which may be formed by small mo-

[2]In this latter case, we do no longer talk of decentralized SIP, but of the standard SIP protocol

bile devices, with low computing capabilities and battery resources. This particular environment pushed us to minimize the signaling exchange needed to register SIP users in ad-hoc networks and to choose a lightweight SIP stack as basis on which to build dSIP.

5.5 SIP Operations in Ad-Hoc Networks

We now pass to the description of our solution: the rest of this chapter is dedicated to discussing various aspects of decentralized SIP. The first important concept is to underline that SIP operations in ad-hoc networks differ from operations in networks assisted by servers, as all the nodes have to rely on themselves to find other SIP users and to contact them.

In brief, SIP operations in ad-hoc networks can be divided into two steps:

1. Discoverying users currently available in the network (Registration)
2. Initiating and managing sessions with them

The user discovery phase in ad-hoc networks roughly corresponds to the registration phase in infrastructured networks, where SIP users register their bindings at their predefined registrar. User discovery is particularly important in ad-hoc networks, since users cannot always expect to simply pick one contact from their list and begin a session, as they would do in an infrastructured network. In an ad-hoc network, in fact, users do not usually know a priori which users are present. A SIP solution in ad-hoc networks must provide means for initiating sessions with previously unknown users as well. Initiating a SIP session is not possible if at least the AOR of the invited party is not known in advance. Decentralized SIP allows users to discover the identity of other nodes in the network; the information discovered includes the AOR of the users on-line, and the associated IP address where they are currently reachable. In other words, during the registration phase the bindings of the other ad-hoc users are discovered and the user's own binding communicated to the other nodes.

We have identified three typologies of users of decentralized SIP, based on the registration related behavior in the ad-hoc network. Some users, which we call the open users, want to communicate their presence in the ad-hoc network to everybody. The second category of users, referred to as the closed users, desires to only communicate their contact information to specific users, for example, those in their local contact list. The third category is middle ground between the previous two; it is formed by users who want to communicate their information to everybody, but wish to be notified only of the presence of a particular subset of users. To the hybrid category belong also users who wish to be notified about the

presence of everybody in the network, while communicating own contact information to selected users only. Decentralized SIP provides means for addressing the needs of these three categories of users.

5.6 Decentralized SIP in a Nutshell

Decentralized SIP is essentially a distributed SIP location service for ad-hoc networks, or more generally, for distributed environments where no support from centralized servers is available, needed or desired. The main goal of decentralized SIP is to allow a user to advertise his SIP contact information, the binding, and to retrieve the bindings of others users in the network. Once a node has resolved into an IP address a SIP AOR in an ad-hoc network, session management operations follow the SIP specifications. Decentralized SIP, being a middleware solution, is orthogonal to any underlying ad-hoc routing protocol: any protocol can be used to route unicast decentralized SIP requests and responses to the intended recipient.

As mentioned before, decentralized SIP is formed by three building blocks: fully distributed SIP or dSIP is the core of decentralized SIP, and enables the distributed SIP location services using SIP-only methods. SLP-aided SIP, or sSIP, was designed as an alternative solution to dSIP, and is more suitable in environments with limited bandwidth availability or where more security in the user discovery operations is desired. The price to pay for these feature is a possible loss of interoperability, since sSIP uses a third party protocol to perform user discovery operations. The last building block is an extension to decentralized SIP, which we call SIPCache, allowing its deployment in big multi-hop ad-hoc networks.

These three building blocks have been presented in several publications. The conference paper [67] discusses the operations of dSIP, while the paper [66] focuses more on the role of sSIP and on the service framework of which decentralized SIP is part of (see Section 1.6). The most comprehensive publication is the journal paper [56], which extends the previous two articles presenting a more detailed description of dSIP, sSIP, and especially focuses on their interactions. It also introduces the security properties of decentralized SIP and presents an evaluation of the protocol overhead. The multi-hop extensions are presented in a range of publications. The work for extending decentralized SIP to multi-hop networks is currently in progress, and the algorithm which is at the basis continuously evolving. SIPCache builds on top of PCache, an algorithm for distributing and retrieving data items in ad-hoc networks. PCache is discussed in [6, 81, 82]. SIPCache, which adapts PCache to the operational environment of decentralized SIP, has been presented in the paper [69].

5.7 Triggering the Working Mode

In ad-hoc networks, as well as in infrastructured networks, SIP users are identified by their AOR. We modify the SIP AOR of a user, say sip:user@domain.dom, so that in ad-hoc networks, it has the form:

sip: user@domain.dom.local

The user does not need to change his SIP AOR when using decentralized SIP: the .local extension is added by the decentralized SIP software stack (the software architecture is discussed in Chapter 6) at the end of the AOR and is used to univocally identify a SIP user in the ad-hoc network and to trigger the working mode. For example, if the target Request-URI in a SIP request has the .local extension, it means that the request is addressed to a user in the ad-hoc network.

The addition of the .local extension is not the only way to trigger the working mode in the server and user agent modules. For example, if the loopback address is used as the SIP target-URI, as is the case when the user agent sends a REGISTER message to the local server (see later Section 6.3), ad-hoc operations are triggered as well at the local server.

Another way to trigger the working mode is based on the IP address; if the IP addressing scheme used in ad-hoc networks is not global, and addresses are chosen from a dedicated subset (such as 169.254/16 as suggested in [91]), the device can know whether it is operating in ad-hoc mode or not. The IP address-based scheme cannot be utilized if the ad-hoc IP addresses are globally routable, since the knowledge of the IP address only would not give the device knowledge on the current operational environment. Such a scenario would be possible in ad-hoc networks connected through a gateway to external, infrastructured networks. In this case, other information can be used to trigger the working mode, such as, whether the WLAN card of the device is working in ad-hoc mode. In the rest of this dissertation, we assume for the ad-hoc network a private addressing space, as this scenario is much more challenging from the interoperability with infrastructured networks point of view.

In other words, by merely looking at the content of a SIP message, it is possible to decide which operations to perform: this design choice allows the decentralized SIP software stack to be transparently reusable both in ad-hoc and infrastructured networks.

5.8 Fully Distributed SIP

Fully distributed SIP is the main of the two operational modes defined for decentralized SIP. We distinguish two types of operations for dSIP, proactive and

reactive. In proactive mode, user discovery is made when starting a SIP application, and registration information is refreshed periodically. In reactive mode, user discovery is performed only when a user wants to initiate a SIP session.

5.8.1 Proactive dSIP

When a user in an ad-hoc network starts his SIP client, a SIP REGISTER message is sent to the local server, which is running in the same device as the user agent. The message contains the user's binding. The server can take different actions based on the local user's policy. For open users, it sends to the network a REGISTER message containing the binding of the registering user. Closed users do not spread their contact information. Sending to the network the REGISTER message serves the double purpose of advertising a user's own binding to other nodes in the network, and triggering replies from the other nodes, containing the ad-hoc users' bindings.

The way of spreading the message in the network is transparent to dSIP; a message can be broadcast, multicast, flooded, or a data dissemination algorithm can be used for the scope. The type of message spreading technique depends on the type of network: in small proximity networks, where all the nodes are within direct contact with each other at link layer, broadcasting is the best option. For small multi-hop ad-hoc networks, multicasting[3] (provided that the nodes run an underlying multicast ad-hoc routing protocol) or flooding can be good choices. In big multi-hop ad-hoc network, running dSIP on top of a dedicated data dissemination algorithm is another alternative, and is also the solution we have chosen to extend dSIP to big multi-hop ad-hoc networks.

In small networks, introducing sophisticated algorithms for spreading the initial REGISTER would add unnecessary computational complexity to the scheme; our approach is an optimal trade-off between scalability and complexity, given the operational target network environment. Comparing the efficiency of dSIP over various message spreading techniques is however out of scope of this dissertation, which focuses more on the middleware aspects of session management. Since the multi-hop case will be thoroughly studied when discussing PCache and SIPCache, we will describe for simplicity dSIP (and sSIP as well) operations in a link-local environment, assuming that messages are spread in the network using broadcasting.

When the nodes currently present in the ad-hoc network receive a broadcast REGISTER, they can reply with a SIP 200 OK message, containing the binding of the replying user. Particularly, open users reply to any received REGISTER message; closed users reply only if the AOR of the registering user is locally known,

[3]The SIP specification defines a multicast IPv4 address (224.0.1.75) for registrations

e.g., it is comprised in their contacts list. Hybrid users always send a registration message, but specify in it the AORs of the users from which they desire a reply. This behavior is justified by the fact that some users want to communicate their information without receiving a high number of reply messages. The evaluation chapter (Chapter 10) shows in fact that the overhead of dSIP is mainly due to the replies to registration messages. For sake of generality, we for now on refer to operations with open users. In this case, after the initial REGISTER - 200 OK exchange, dSIP nodes have the knowledge of all the other users' contact information in the network. In networks of open users, dSIP allows complete knowledge of the other SIP users's bindings in the network with minimal signaling exchanges and protocol complexity, which meets one of the design goals of the solution.

Decentralized SIP differs slightly from baseline SIP because with the 200 OK message standard SIP registrars return to a registering user the user's own contact information as confirmation of successful registration. In dSIP, the bindings of the replying users are returned instead. According to SIP, bindings have a limited life time. Registration messages should therefore be periodically refreshed to avoid bindings expiration. Refresh messages are also needed to cope with losses of the original message. Even if a user is open, proactive dSIP does not reply to a refresh REGISTER, in order to save bandwidth and battery resources. Each node keeps the ID and sequence number of the received and sent REGISTERs. Refreshed registrations have the same ID but higher sequence number than the original or the previous ones; in this way nodes can discriminate whether the REGISTER is original or a refresh, and avoid answering with a 200 OK message. If the REGISTER message is a refresh, nodes simply update their cache entry.

If information about available users is present in the local server cache, users can begin SIP sessions. When a user wants to initiate a session, a request is sent from the user agent module to the local server module for the list of users currently available in the ad-hoc network. The query issued by the user agent is carried in a SIP MESSAGE request [16]; the user list is returned in the body of another SIP MESSAGE that the local server sends to the user agent module. These messages are exchanged inside the same device, and therefore they do not waste bandwidth.

The MESSAGE extension to the SIP protocol provides support for paging-mode instant messaging; for dSIP purposes it is very useful as it allows to issue non standard requests and receive answers. We have used MESSAGE in dSIP only for requesting and returning the user list. However, the method is flexible enough and it can be used to realize other functionalities. The use of the MESSAGE method was also suggested by the effort of limiting modifications and additions to the baseline SIP protocol. In dSIP, no additions to the baseline SIP protocol (new messages or header fields) have been defined, and the messages respect the syntax of the protocol. Only the semantic has changed in some cases; e.g. the

format of some header fields triggers specific actions at the server. This approach allows interoperability with the baseline SIP specification [67].

When the user has seen the list of peers currently available, he can invite one of them to a SIP session. The INVITE message is forwarded to the local proxy server, and then to the server module of the final recipient node. From here, it will be forwarded locally to the intended recipient. The logic of session establishment of SIP is respected; the only difference is that inbound and outbound proxy servers involved in the session establishment process are now co-located with the user agents. Next chapter will explain in more details how dSIP software components can trigger ad-hoc or standard operations, e.g, how they can trigger routing a SIP message to the local server instead of the predefined outbound proxy server.

When users leave the ad-hoc network, they can send a REGISTER with expiration value set to 0. Similarly to standard SIP, this message triggers all the receiving registrars to remove the binding for the unregistered user. Unregistration messages are not replied to.

5.8.2 Reactive dSIP

The operations of dSIP in reactive mode are similar, as is the message format, except for the timing of messages exchanged. User discovery is triggered by the user (instead of being automatic when the SIP application is started) when sending a MESSAGE requesting the user list. The server receives the MESSAGE and sends a REGISTER, waiting for other nodes to reply with a 200 OK. Same considerations on the type of users (open, closed, hybrid) done for the proactive version apply here, and we still refer to open users operations. After some preconfigured time has elapsed, the server returns the list of SIP users to the querying local user agent with a MESSAGE. Session invitation steps coincide in the two approaches.

In reactive mode, no refresh registrations or unregistrations are sent, as the philosophy behind this approach is to send messages only when needed by the querying user. This implies that the cache entries in the server are not kept updated as in the proactive case. For example, nodes do not notice leave events from other nodes, and this can result in unsuccessful invitations if a node relies on old cached information. Therefore, a REGISTER should be sent every time the server receives a session invitation, even though there are entries in the cache that are not expired. Refresh REGISTERs are always replied to in reactive mode. A hybrid approach consists of sending the first registration when the user decides to begin a session, and backing off to proactive operations afterwards. That is, registrations should be refreshed periodically and must not be replied to by the nodes that receive them.

5.8.3 Targeted dSIP

The registrations operations described so far are general for all users; when a node sends a REGISTER it does not specify users to which the message is targeted. The same mechanism applies to refresh registrations and unregistrations as well. Nevertheless, there may be situations where a user would like to indicate a specific user to which a registration request is targeted, such as in the case of hybrid users.

This operation mode, referred to as targeted dSIP, is not the default one because generalized registrations are enough to fill nodes' caches with information about the users in the network. An example of use of targeted dSIP is when a user knows that a friend is in the ad-hoc network, but for some reason he did not get his contact information. He can send a refresh broadcast or multicast REGISTER, specifying the friend's AOR as target request-URI of the message. The other nodes in the network do not reply to the message, as it is a refresh one, except for the user specified in the target Request-URI, if present in the network. In generalized (proactive or reactive) registrations, the request-URI of the message is the broadcast address for the ad-hoc addressing space in use. In targeted dSIP, it is the target contact AOR, with the .local extension. Both types of URIs trigger ad-hoc operations in the dSIP software stack.

5.9 SLP-Aided SIP

SLP-aided SIP, or sSIP, is basically a proactive version of dSIP, where registration operations are performed with a service discovery protocol, rather than with SIP-only methods. The consideration on closed and open users made for dSIP are valid for sSIP as well. The support of a service discovery framework is useful in ad-hoc networks to give users the possibility to discover people, services, or devices in the network. For example, one of the devices in the network may have Internet connectivity, and offer this service to users in the network looking for the service "gateway".

Service discovery can support user discovery in decentralized SIP either by finding out the bindings of users within reach in the ad-hoc network or to discover the IP address of a user using the SIP AOR as query filter. This latter method is correspondent to deploying the SIP location service with the standard protocol means. Various service discovery frameworks can be used to support decentralized SIP. The device needs only an API so that the SIP modules and external applications can issue service discovery queries. Among the service discovery frameworks that could fit our architecture (for an overview of service discovery standards, see Helal [44]) we have chosen the Service Location Protocol (SLP) [132]. The reasons are that SLP is an IETF Proposed Standard, published already in 1999 and therefore mature and widely understood, also in terms of available

open source implementations. Additionally, SLP is quite lightweight and fits well the needs of small mobile devices. The most important factor that motivated the choice is that SLP can be easily adapted to use in a decentralized environment, like ad-hoc networks. We did not adapt SLP for use in ad-hoc networks, but used an implementation realized at the Technical University of Tampere as part of the SESSI project, which provided the research backbone of this dissertation. More details on how SLP was adapted for use in ad-hoc networks are presented in the SESSI common journal paper [56].

5.9.1 Overview of Operations

Since SLP substitutes SIP in performing registration[4] operations, it must give ad-hoc nodes the possibility of retrieving the bindings of SIP users in the network. This means that the current IP address and the SIP AOR should be returned when performing an SLP query. The IP address is easily retrieved as the location of the queried service. The AOR is treated in SLP as the service attribute. Table 5.1 shows the mapping between the REGISTER header fields used in SIP and the SLP service request fields.

Table 5.1: Header fields mapping SIP - SLP for ad-hoc registrations

REGISTER	**SLP**
Contact	Service Address
AOR	Service Attribute

When SLP is used in the registration, the location service can be deployed by broadcasting SLP service request messages. The query is meant for the nodes running the service *SIP*. All the nodes that have locally registered the service SIP to the SLP service discovery framework reply by sending the address where the requested service type is available, that is, their IP address and the user's AOR as SLP service attribute. The reply from the other nodes allows the server module to construct the bindings of the users in the network. Once the bindings have been retrieved, the operations of dSIP and sSIP coincide, and session management is handled in a similar way. With SLP, it is also possible to perform the equivalent of dSIP targeted registrations, by sending a query for the service SIP, containing the AOR of the user to contact as attribute filter. All devices in the ad-hoc network receive this request and the one that matches the attribute AOR returns the IP address of the service SIP on that host. When the server module receives the response, it stores the IP address of the service in its cache.

[4]For SLP, we could more properly refer to user discovery phase

5.9.2 Qualitative Comparison of dSIP and sSIP

User discovery with SLP has some advantages over decentralized SIP: the most important is that SLP consumes less bandwidth than SIP, since its messages are binary encoded. Section 10.4 provides an analysis of the bandwidth consumption caused by the two approaches. A disadvantage of using a service discovery framework lies in interoperability issues. If all the devices in the ad-hoc network use the same service discovery framework, then it is possible to retrieve the bindings of all the users; otherwise, if devices use different and incompatible service discovery frameworks, retrieval of users' bindings may be impossible.

A major difference between dSIP and sSIP is that with broadcast SIP REGISTER messages, receiving nodes can store the bindings of the registering user. When an SLP query for the service "SIP" is received, nodes reply but do not get the binding of the registering user for storing. This implies that when SLP is used, new nodes in the network get to know the bindings of older nodes, but the opposite is not true.

There are two solutions for this problem: one is to refresh periodically the broadcast SLP query for the service "SIP", so that if new nodes have arrived in the interval between two refreshes they can communicate the bindings. Another option is enabling "SLP Passive Service Discovery". This is an extension to the baseline SLP protocol, which allows nodes entering the network to advertise their service, by broadcasting an SLP message that contains the bindings upon connection to the network. The message contains the type of message advertised ("SIP" in this case), the address where it is reachable and the user's AOR as attributes. The scheme is called passive, as nodes passively receive information about active services in the network directly from the service providers. Passive service discovery extensions have been implemented by Tampere University of technology within the SESSI project; adapting decentralized SIP to passive SLP service discovery is subject of future work.

A third alternative would be to modify the SLP working logic so that, when requesting a service address and its attribute, it is possible to include in the query message the querying node service address and attributes. With this solution, when ad-hoc nodes receive an SLP query, they can store the data of the querying node. However, extending SLP for achieving passive service discovery or storing the data of the querying node is out of the scope of our work. However, this solution does not fit well with design goals of minimal extensions to baseline protocol, as it requires changes to existing SLP message, with possible loss of interoperability with standard implementations.

5.10 dSIP for Big Multi-hop Ad-hoc Networks

The rest of the chapter presents an algorithm devised to adapt decentralized SIP to big multi-hop ad-hoc networks. As a reference, the decentralized SIP version used to show the extension, is proactive dSIP. The algorithm can be transparently applied to any version of our session management solution, since it is a routing layer algorithm, sitting below decentralized SIP middleware layer and meant for optimizing general message spreading techniques, such as flooding.

We underline that big multi-hop ad-hoc networks are not the most suitable operating environment for real time applications, such as voip or video calling. These applications have strict timing requirements, and meeting them presents an interesting challenge in a dynamic environment such as ad-hoc networks, due to the unreliability of any routing scheme and the varying network topology. Nodes mobility adds to the complexity of the scenario. For practical reasons, we do believe that real time communication can and must be achieved also in completely peer-to-peer, serverless fashion, and that small (proximity) networks are an environment where this goal can be achieved.

In this dissertation we propose a solution for the signaling part of real time applications, while solving the problem of achieving real time communication in multi-hop ad-hoc networks is out of scope, although a possible candidate future research topic. The solution we present ideally complements the core of our session management framework, and makes it complete to be deployed in any network environment.

5.10.1 Problem Statement

The main dSIP architecture is optimized for proximity networks, as it minimizes the signaling exchange needed to exchange bindings in the network and does not utilize complex message spreading techniques to distribute messages to the other network nodes, taking advantage of the nature of the operating transmission medium, wireless link-local networks.

Another assumption of dSIP is that, in theory, in network of open users, all the users can have knowledge of the bindings of all the other users, storing this information locally. This assumption must be revised when adapting the algorithm to bigger networks, of possibly several hundreds of nodes. The problem is not in the limited storage space of nodes, as a binding occupies only a few bytes of memory, and even a hundred of them would amount up to only a few Kb of memory usage. Rather, the issue is in the efficiency of the scheme, also in terms of signaling: it is not reasonable to design the binding retrieval mechanism such that each arriving node should receive a reply message from each node in the network. The amount of traffic generated by this scheme would be unbearable, also because the routing

algorithm used to forward unicast replies to the registering node would trigger a storm of broadcast route discovery messages. The wireless medium would be occupied only by routing messages, with little bandwidth left for signaling messages, and even worse, for actual data traffic. This observation has been confirmed in some preliminary tests we have run: the simulations referred to a network of a hundred nodes, where each registering node broadcasts a registration message and receives a unicast reply from the other nodes. The simulation showed that the traffic in the network was almost fully due to route discovery messages for the unicast replies.

The solution we have designed to this problem relies on the broadcast nature of the wireless medium, and minimizes the number of unicast messages sent in the network. We use a modified version of gossiping algorithms, where nodes broadcast a received message based on probabilistic criteria, and leverage on cooperative caching to achieve a distribution of bindings in the network as geographically even as possible. The aim is that a reply to any binding request message can be returned to a querying node in the minimal number of hops (and hence, overall exchanged messages) possible.

5.10.2 An Information Management Algorithm for Ad-Hoc Networks

PCache [82, 6, 81] is, in brief, an algorithm for information management in ad-hoc networks, including data dissemination and efficient caching. Information management in wireless ad-hoc networks is not a straightforward task. The inherently distributed nature of the ad-hoc environment, and the dynamic characteristics of both network topology and medium connectivity, are challenges for the efficient handling of data. There are several policies that can be followed when dealing with information management in ad-hoc networks. In particular, one must first choose whether to follow a centralised or decentralised approach.

Due to the nature of the ad-hoc environment, solutions that concentrate the data in a single entity must be discarded; the node hosting such a centralised entity would require significantly more resources than other nodes. Usually, ad-hoc networks are formed by peer nodes, with limited capabilities, so it is not practical to elect one node to act as a repository for the data needed by the other nodes. Moreover, this approach introduces a single point of failure; this is unacceptable given that node failures are an integral part of ad-hoc networks (failures may happen due to voluntary departures, crashes, or simply due to medium impairments). A decentralised approach is, therefore, strongly favoured and coherent with the philosophy of our session management framework.

In a decentralised approach, the data items are spread among all the nodes of the network. The data dissemination algorithm should balance the need to provide data replication (to cope with failures) with the need to avoid excessive data

redundancy (as nodes may have limited storage capability). Furthermore, data items should be distributed as evenly as possible among all the nodes forming the network, avoiding clustering of information in sub-areas; even dissemination of data items should leverage lower access latency to any item from any node in the network. An even distribution of information in the network implies that whenever a data item is requested by a node *S*, the distance to the node that provides the reply is approximately the same, regardless of the location of *S*. Naturally, the actual distance depends on multiple parameters, such as the number of nodes in the network, the size of the cache where data items are stored, and the number of data items. From a latency point of view, one should aim at minimising distance (i.e., ideally, any data item should be available from one of the 1-hop neighbours of *S*). Finally, since in wireless ad-hoc networks both bandwidth and battery power are precious resources, the algorithm should also minimise the amount of signalling data.

We have designed the PCache algorithm based on these goals. The algorithm provides two separate operations: dissemination and retrieval of cached data items. The implementation of these operations is orchestrated such that a limited number of messages is required for retrieving any data item from the network, independently of the addition, removal or movement of nodes. This goal is achieved by a combination of four different complementary mechanisms: an efficient best-effort probabilistic broadcast mechanism; a distributed algorithm for deciding which nodes replicate a given data item; a data shuffling mechanism to improve the distribution of data replicas and an expanded ring-search mechanism to support queries.

There are several interesting applications for our algorithm. In the context of our session management framework, we use PCache combined with dSIP to efficiently distribute and retrieve SIP user bindings in ad-hoc networks. We call the joint solution SIPCache.

PCache is a continously evolving algorithm. In this dissertation we present the original version of it; however, work is currently in progress to enhance it and fine tune its parameters. At the time of writing, two more articles on PCache, besides those already published, have been submitted to conferences and are pending review.

5.11 The PCache Algorithm

Each node in a PCache system has a cache of a limited and predefined size. The cache is used to store a fraction of all the data items advertised. Each data item is composed of a key, a value, an expiration time and a version number with application dependent semantics. Nodes continuously pursue a better distribution of the

items, by varying the content of their caches. The goal of PCache is to provide an adequate distribution of data items so that each node is able to find a significant proportion of the total items in its cache or in the cache of the neighbours within its transmission range. Items in PCache are distinguished into those owned (that is, advertised) by a node, and so called complementary items, which are items held by a node, but not owned (that is, items advertised by other nodes), whose usage in PCache will be explained later.

PCache is a reactive protocol in the sense that it only generates packets to satisfy the requests of applications. PCache provides two distinct operations: data dissemination and data retrieval. These operations are implemented using three types of messages. In the dissemination process, nodes cooperate to provide an adequate distribution of the replicas of new or updated versions of data items. *Dissemination messages* are broadcast following an algorithm to be described later. The retrieval process is triggered by applications requesting to PCache the value associated with a key. The protocol first verifies if the value is stored in its local cache and if it is not, it broadcasts *query messages*. Nodes having in their cache the corresponding value address a *reply message* to the source of the query. *Data Gathering* messages are a particular type of messages used to perform queries for items satisfying a particular set of conditions, unlike query messages that are used to retrieve the value of an item indicated by its key as search parameter.

This section begins by presenting the structure of the cache at each node and the content of PCache messages. It then provides a description of the dissemination and retrieval processes and of the handling of the Complementary items.

5.11.1 Cache Structure

Nodes in a PCache system provide storage space for caching a fraction of the items advertised by all nodes. Nodes always try to keep their caches full, occupying all free space before beginning to overwrite other cache entries. The system does not require the caches at all nodes to be of the same size but proposes a common format and cache update policy. Each data item is stored in cache together with auxiliary information to support an adequate distribution of replicas (see Fig. 5.1). The popularity ranking counts the number of times that a node listened for the item in the messages received previously. The *eraseCandidate* flag helps to leverage item distribution by suggesting the items that are more adequate for replacement.

A data item is said to be owned by the node that initiated its advertisement after an application request. To ensure that at least one copy of each item exists, nodes do not replace the items they own with items advertised by other nodes. It is assumed that owned items are stored in a separate region of the memory space of the devices so that the space available for caching third-party records is kept constant.

```
cached item {
    type: {owned,remote}
    key: opaque
    value: opaque
    expiration: time
    version: int
    popularity: int
    eraseCandidate: bool
}
```

Figure 5.1: Structure of PCache items in cache

5.11.2 Message Content

PCache messages share a common header that describes the type of message (dissemination, query or reply), a *time to live* (TTL) field, decremented by each node that forwards the message, and additional information concerning the items it carries and their relation with the state of the cache of other nodes. The content of PCache messages is presented in Fig. 5.2.

The fields *source*, containing the address of the node that created the message and *serial number*, containing a number local to each node and incremented at every message it creates, are used to uniquely identify a PCache message. To identify duplicates, nodes keep a record of the messages recently received. In PCache, it is common to have messages to be forwarded and changed by multiple hops. We define the source of a message as the node who created it and defined the value for the *source* and *serial number* fields. Messages are edited and forwarded by multiple *senders* which are not allowed to change the content of these fields.

Items are stored in an application dependent format, transparent for PCache. Similarly to some routing protocols ([52]), query messages accumulate the path to be used by a reply in a field, here identified as *route stack*. The header also carries information to help to leverage the distribution of items. This is the case of a field named *time from storage* (TFS).

5.11.3 Broadcast Algorithm

A broadcast algorithm is used for forwarding dissemination and query messages, although with some differences, highlighted in the respective sections. The basic version of the algorithm, is presented in the technical report [82], while the paper [81] describes an enhanced version of the algorithm, with a more detailed evaluation. The algorithm does not intend to deliver the messages to all nodes

```
type: {dissemination,query,reply}
time to live: int
source: address
serial number: int
time from storage: {0,1,2}
route stack: address[]
# items
items: [] {
    key: opaque
    value: opaque
    expiration: time
    version: int
}
```

Figure 5.2: Content of a PCache message

with high probability. Instead, the retrieval of a data item from the network is guaranteed by the combination of both the dissemination and retrieval procedures and by the replication of the data items. This allows using an unreliable broadcast algorithm, focused on the reduction of the number of messages. The broadcast algorithm puts together a mechanism to limit the number of retransmissions in floodings (similar to those in the papers [42, 87]) and a protocol that use the signal power of a received message to optimise further message flooding [70, 87]. It is assumed that the reception power of a message can be provided by the network card driver.

The algorithm tries to reach the biggest number of nodes with the lowest number of transmissions. Therefore, the algorithm privileges retransmissions performed by nodes located farther away from the previous sender, which have a higher probability of reaching a bigger number of the nodes that have not yet received the message. To limit the resource consumption of nodes, the algorithm also prevents from retransmitting nodes whose contribution to the number of nodes covered is believed to be small.

For each message m, the broadcast algorithm works as follows. Each node receiving for the first time a copy of m will place it on hold. The hold time is proportional to the power with which the message was received. Disregarding any fading effects in the wireless media, it is expected that nodes more distant to the sender of m have a lower holding period. During the hold period, each node counts the number of duplicates of m it receives. Preliminary simulations showed that in the majority of cases, a node listening to at least two retransmissions can

discard the message without negatively influencing the message dissemination expectations. Therefore, a node will retransmit *m* if it has listened to less than two retransmissions of the message. The message will be marked for dropping otherwise. The handling of a message is dependent of its type and further described in the following sections.

5.11.4 Dissemination Process

The rationale for the dissemination process is better explained assuming a configuration where nodes do not move, and that nodes have unlimited cache size (so that entries in the cache are never replaced). In this scenario, the dissemination process provides a reasonable probability that all items are found within the transmission range of every node. However, even in situations of limited cache size and nodes mobility, the algorithm provides a reasonably even distribution of data items, as Section 10.6 will show. The dissemination algorithm mandates that, starting at the last storage, every third node propagating a dissemination message stores the advertised items. Complete determinism is removed from the algorithm by permitting other intermediary nodes to store the record, although with a small probability. In principle, this approach allows any intermediary node to have a copy within its 1-hop neighbourhood; the copy would be located either in the node from which a message was received, or in the next hop (if it exists).

Dissemination of data items is triggered by the source node with the broadcast of a dissemination message. In dissemination messages, the *time from storage* (TFS) field indicates the distance (in number of hops) from the sender to the closest node that is known to have stored the items. Therefore, the source node sets the TFS field to zero to indicate that the records are stored in its local cache.

Each node receiving a dissemination message places it on hold for a period of time proportional to the reception power, as described in Section 5.11.3. During the hold period, the node counts the number of retransmissions listened and calculates mintfs, which is the lowest value of the TFS from the original message and of all retransmissions. At the end of the hold period, mintfs will indicate the distance in hops from the source to the closest node(s) that stored a copy of the item. When the hold period expires, the node uses the number of retransmissions listened, mintfs and a random number generator to decide for one of three possible actions:

- If a node listens to two or more retransmissions and mintfs is 0 or 1, following the rationale of the broadcast algorithm, it can safely discard the message. Listening to two or more retransmissions suggests that the propagation of the message in the neighbourhood is being assured through some of the neighbours, so there is no need to further forward the message. A

low value of mintfs (0 or 1) indicates that a close neighbour has stored the message, so it is advisable to reserve space in the cache for items carried in another message.

- The data item is stored in the node's cache and the message is retransmitted. This will be the action to execute with probability $e^{mintfs-2}$ if the first criterium did not apply. The probability of storing an item increases with the distance to the closest copy. In particular, if the closest copy is three hops away (signalled by a mintfs of two) a copy will always be stored in the node. Nodes executing this alternative will forward the dissemination message with the TFS field set to 0. As a consequence, the mintfs of neighbour nodes that did not terminated the hold period will be set to the lowest possible value and will have their probability of storing the item reduced. From the above, it can also be concluded that TFS and mintfs are always bound between 0 and 2. PCache benefits from having some randomisation associated with the decision of storing an item. $e^{mintfs-2}$ has shown to be adequate because it presents an exponential growth with the distance to the closest copy. The probability of storage for nodes with mintfs of zero or one is respectively 0.14 and 0.36.

- A message will be forwarded but the data will not be stored in the cache if none of the previous conditions applied. The TFS of the retransmission will be set to mintfs+1 to inform the listening nodes of the additional hop to the closest node that stored the item.

Fig. 5.3 exemplifies a dissemination where any node is able to retrieve the item in its 1-hop neighbourhood. Nodes that forwarded the message are represented in gray and nodes that stored and forwarded the item in black. Three copies of the item were stored. The first at the source node (Fig. 5.3(a)), the second for probability (Fig. 5.3(c)) and the third because the node had a mintfs of two (Fig. 5.3(d)). For clarity, only a subset of the message receptions are represented.

5.11.5 Retrieval Process

The retrieval process in PCache is triggered by the request for a data item from the application using PCache. When a request is received, a node begins by looking for the key in its local cache. If the value is not found, the node prepares a query message, placing the key in the message. An expanding ring search is performed due to the expectations that the dissemination process was able to store the value in the 1-hop neighbourhood of the node. The message is first broadcast with TTL equal to one. The query is reissued with a large TTL if no reply is received within

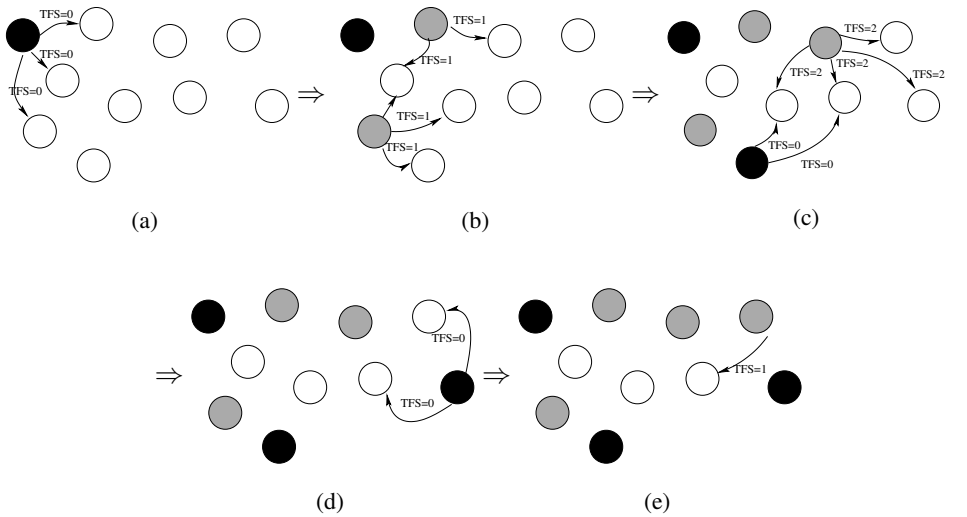


Figure 5.3: Progress in dissemination of an item

some predefined time limit. The protocol imposes a limit on the number of retries to be performed, which occur at growing time intervals.

A node receiving a query message, and that does not find the value in its local cache executes the broadcast algorithm described previously. The message is retransmitted if TTL permits and the node does not listen to the broadcast of two copies of the original query. Prior to broadcasting, the sender appends the address of the previous hop to the *route stack* field, in a process similar to the route discovery algorithm in some source routing protocols for MANETs, like DSR [52].

If a node receiving a query message finds the requested key in its cache, it does not enter the holding period of the broadcast algorithm. Instead, it sends a point to point reply to the source of the query. The reply message is sent after a random delay, to prevent the collision of multiple replies. PCache makes no provision to limit the number of replies addressed to a node. Therefore, one reply from each node listening to the query and with a replica of the item in its cache can be received. The expanding ring search is expected to limit the number of replies in the majority of the cases. When the probability of finding an item in the 1-hop neighbourhood is low, PCache can be adapted to reduce the number of query floods and therefore, reduce the number of replies, by slightly changing the expanding ring search procedure. The TTL for the queries can start at a higher value (e.g. 2) to cover a larger number of nodes in the first round or can increase

progressively until a reply is found. A comparison between TTL initial values of 1 and 2 is presented in Section 10.6.7.

The path constructed in the query message identifies a sequence of hops that were unable to provide the requested value. The reply message follows this path, with each intermediate hop addressing it to the one that preceded it in the query propagation. The item is stored at the node that issued the query. It should be noted that PCache is orthogonal to the underlying routing protocol. If no ad-hoc routing protocol is available, PCache unreliably broadcasts the message to the network without verifying if the destination is still in range. The reply messages contains the information on the chain of hops the query message has formed before arriving to the node generating the message. So, a node receiving a reply message, no matter if broadcast or unicast via an underlying routing protocol, knows the next hop which is supposed to process the reply.

The use of a routing protocol for MANETs for delivering reply messages would increase the network traffic as a flooding could be required to find a route between the replying node and the source of the query. On the other hand, PCache makes no provision to limit the number of replies sent to the node. Therefore, there is a reasonable probability that at least one of the routes constructed during the query propagation remains valid until the reply is delivered. It should be noted that this assumption is similar to that of many reactive routing protocols for MANETs [52, 90] for route discovery.

5.11.6 Data Gathering Mechanism

The Data Gathering mechanism is used to retrieve more than a single item within the same query message. The query is performed by specifying some condition that must be satisfied by the keys. Data Gathering operations may produce a large number of redundant messages in both the dissemination of the query and on the forwarding of the replies with obvious implications in the resource consumption of the devices. As a preliminary step to decrease redundancy, PCache relies on the efficient replication of data items to impose a limit on the distance (in number of hops) that the query travels by performing a ring-search. The radius of the ring is defined at the source of the query in a field commonly identified as Time-To-Live (TTL). Each node forwarding the message decrements the value in this field. The message ceases to be forwarded when TTL reaches zero.

The nodes in the ring are dynamically partitioned in clusters. Each cluster head is responsible for decreasing redundancy by aggregating replies from the remaining cluster members. Redundancy is reduced by preventing nodes from sending replies containing data items already known by the cluster heads. The main characteristic of our clustering approach is that it is stateless and message-driven, that is, clusters are formed based only on the content of received messages.

There is no need to implement a dedicated membership protocol, or for the cluster heads, to keep state information on the nodes belonging to their cluster, similarly to the general PCache broadcast algorithm.

A Data Gathering message is broadcasted by a source node *S* containing a description of the query and the records present in *S*'s cache that satisfy it; in fact, a condition can be satisfied by more than one item. These records are added to the Data Gathering message to prevent redundant replies. All nodes become members of the cluster headed by the node from which they first received the query message. Therefore, all nodes in the transmission range of *S* become members of its cluster. A node creates and becomes head of another cluster if the TTL of the message permits its retransmission and if it decides to retransmit the message.

The policy for making such a decision is the following. Before retransmitting, the cluster head decrements the TTL and appends its address to the list of forwarders in the message. The chain of cluster heads that is formed defines the route to be followed by the reply messages. The node also searches its cache for data items satisfying the query but that were not present in the incoming message. These records are appended to the gathering query message up to the message maximum size and to a reply message, that is placed on hold for a period of time proportional to the value of the TTL field in the message. If the size of the records satisfying the query exceeds the maximum message size, a random subset is selected. A node deciding not to become a cluster head sends a reply with the data items in its cache satisfying the query and that were not present in the gathering message or in any other retransmissions of the Data Gathering message listened by the node while on the waiting period. No message is transmitted if the node does not have any data item satisfying the above conditions in its cache. The reply message is addressed to S and follows the route advertised in the query. Therefore, it is first delivered to its cluster head. The reply may consist of several messages if the data items satisfying the condition do not fit the maximum message size for the network. It should be noted that since messages follow an hop-by-hop route determined at the PCache level, the gathering algorithm does not depend from a routing protocol.

A cluster head receiving a reply addressed to S and that is still on its hold period aggregates the data by appending the non-redundant data items to its reply. When the timer expires and if the reply is not empty, it is sent, addressed to S but delivered to the previous cluster head in the chain started at S. Eventually, all replies reach node S. To save resources, cluster heads discard all state concerning the query after sending the replies. Thus, all replies received after the expiration of its timer are forwarded by the node without further processing. The reply may be aggregated on one of the other cluster heads in the path to node S if its timer has not expired or be delivered to S. A run of the algorithm is exemplified in Fig. 5.4.

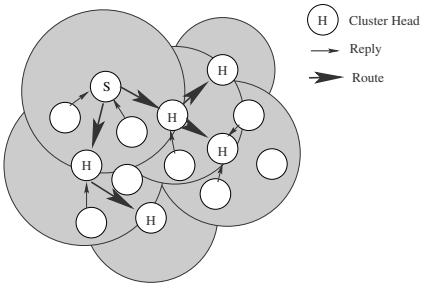


Figure 5.4: Propagation of Data Gathering messages and replies

5.11.7 Complementary Items

In complement to the information relevant for the action in progress, in PCache, all messages carry as many *Complementary items* as possible, without exceeding a predefined maximum message size. The role of Complementary items is to leverage an even geographical distribution of the information by mixing fractions of the cache of different nodes that have been forwarding the message and by letting neighbour nodes learn relevant information about the state of each other caches, without requiring a membership protocol. These items are not relevant for the operation taking place, but help in propagating data throughout the network. Complementary items are handled similarly regardless of the type of the message that carries them.

The transmission of a message has a non-negligible fixed cost that is independent of the message size. In the IEEE 802.11 protocol family, for example, this cost can be attributed, among others, to the fixed size of the MAC and Network level headers and to the contention period. Transmitting the complementary items adds to the size of PCache messages sent over the wireless medium. However, previous analysis [20, 32, 45] shows that bigger packet size in IEEE 802.11-based ad-hoc networks does not linearly increase the power consumption of the mobile devices or decrease the throughput. In PCache case, they improve the replication and geographical distribution of the data. We assume that applications advertise small sized items, so that at least a few are able to fit, together with the PCache and other headers, in a single frame at the link layer level. The behaviour of PCache using messages of different sizes and different number of items is evaluated in Section 10.6.

Each Complementary item carries a flag, *storedInSender*, which is active if the item is stored in the sender cache. When preparing the Complementary items in a new message, the source node places the items it owns and, if there is still space available, the less popular items of its cache (according to the popularity ranking). The *storedInSender* flag is active for all items to indicate that all Complementary

items are present in the sender's cache.

The different operations over Complementary items are presented in Fig. 5.5. In the figure, t stands for true, f for false, pop for popularity, and cand for candidate. All nodes receiving the message update the auxiliary information of the items simultaneously present in their caches and in the Complementary items of the message. These items have their popularity ranking increased. The *eraseCandidate* flag associated with the item in the node's cache becomes active if the item was presented in the message with its *storedInSender* flag on. The rationale is that if some neighbour node already stores a copy of the item in its cache, then a better distribution is achieved if the slot was occupied by some other item. Action a) of Fig. 5.5, for example, sets the *eraseCandidate* flag of item r1 to true, since the item is already stored in at least one of the node's neighbours.

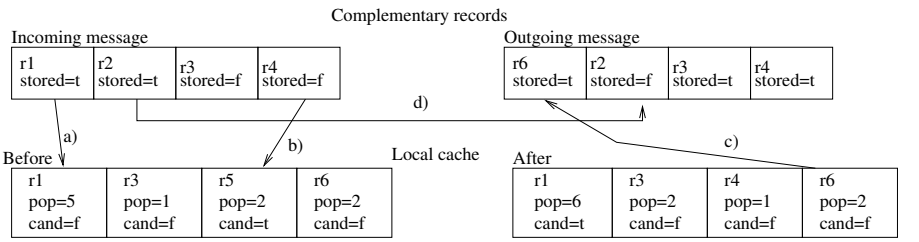


Figure 5.5: Operations performed over Complementary items when forwarding a message.

A node that decides not to drop a message (according to the criteria presented in the previous sections) makes use of the Complementary items to change its local cache. Each Complementary item not present in the node cache and with the *storedInSender* flag off may be stored locally with probability p_{ins} if the cache has space available or any of its entries has the *eraseCandidate* flag active. The item preferably occupies a free slot in the cache. This was the case in action b), where r4 replaces r5, which is known to be stored by some of the node's neighbour.

All nodes forwarding a message can change the Complementary items it carries. Data items owned by a node will replace random positions of the Complementary items. An item simultaneously present in the Complementary items and in the local cache may be replaced in the outgoing packet by the sender with probability p_{rep}. The items to be inserted are those that have a lower popularity ranking in the node's local cache. Action c) shows the case were the forwarded message has the item r1 replaced by item r6, which has the lowest popularity ranking from those not present in the message. Finally, the node is required to update the *storedInSender* flags of all Complementary items in the message so that they reflect the state of its cache. See for example action d), which updates the flag

to false, to reflect the fact that r2 is not kept in the node's local cache and also the remaining items, which have their flag set to true to indicate that the items are present in the local cache.

5.12 SIPCache

This section shows how dSIP and PCache have been combined together to form a functional location service in MANETs, which we call SIPCache [69]. SIPCache extends and optimizes the functionalities of dSIP to large multi-hop ad-hoc networks; in this environment, it is not feasible to assume that each node knows the binding of all the other nodes. Nevertheless, a node should have the possibility of retrieving the bindings of, in principle, all the other users in the network. In other words, the problem is that the SIP location service must be spread among all the nodes in the network since it is not possible to have a centralized repository for the bindings. SIPCache enables distribution of the bindings among all the nodes of the MANET but does not require a node to locally store the bindings of all the nodes of the network.

SIPCache allows distributing and replicating a user's own binding in the MANET. This operation uses the PCache Dissemination algorithm. It is worth noting that while in proactive dSIP own data dissemination and remote users' data retrieval are performed within a single transaction, the REGISTER - 200 OK handshake, in SIPCache the two operations are distinguished: a query for remote user's data must be explicitly triggered by the application. In this sense, SIPCache is a hybrid between proactive and reactive dSIP, since registration (location service maintenance) is done proactively, while queries for remote users data (distributed location service usage) are done reactively, on a need basis.

SIPCache enables the usage of the two types of location service provided by dSIP. The first type consists in retrieving the IP address of a single user based on the user name used as a search parameter. This operation is performed using the PCache Query mechanism. The second type corresponds to the general registration in dSIP, where the nodes may reply to the node that has spread a REGISTER message with a 200 OK message containing their bindings.

SIPCache slightly modifies and enhances this procedure. The enhancement comes from the fact that the registering node may specify the characteristics of the users from which a reply is desired: a user may broadcast his desire to communicate with people interested in sports, or music, rather than art or informatics. This type of queries, which we call profile-based queries, is enabled by the PCache Data Gathering mechanism. The modification deals with how the replies are returned, which is managed by the interaction of dSIP with the PCache Data Gathering mechanism, and with the fact that a general registration in SIPCache is

logically treated as a query. Flooding a query for a user matching a given profile and returning a reply without PCache would imply, as preliminary tests showed, a large number of messages exchanged altogether, due to the high number of nodes that would return a reply. Instead, replying by following PCache algorithm allows saving the number of messages exchanged, which is beneficial from the bandwidth and device resources consumption point of view.

5.12.1 Mapping of SIP Messages to PCache Messages

The interaction between SIP and PCache was achieved by directly mapping SIP messages to the format defined for PCache. SIP messages are distributed into the network according to the procedure defined by the PCache algorithm, and their header fields are processed as the correspondent PCache header field. The correspondence between SIP and PCache messages is shown in Table 5.2.

Table 5.2: Mapping SIP-PCache messages

	SIP	**PCache**
Message Type	REGISTER	Dissemination
	MESSAGE	Query
	200 OK	Query Reply
Header name	Max-Forwards	TTL
	From	Source
	Call-ID	Serial Number
	CSeq	Time From Storage
	Via	Route Stack
	Content-Length	# Items
	Body	Items

A PCache dissemination message is mapped into a SIP REGISTER message. Other SIP methods could have been used for the purpose, but by choosing REGISTER we emphasize the fact that we are spreading a registration in the MANET. The MESSAGE method [16], used for SIP-based page-mode instant messaging, is used for queries, both for retrieving a user's IP address based on his AOR, and for a profile-based query. In the first case, the "Subject" header field of the MESSAGE request will carry the indication "Query"; the AOR of the target user is carried in the SIP Request-URI and in the To header field. In the second case, the indication "Query-all" will be reported; the search parameters are specified in the body of the message. Replies to query are also carried in the body of a MESSAGE request.

5.12.2 SIPCache Registration

A SIPCache registration is spread in the MANET either when a node joins the network, or it is refreshed to avoid SIP bindings expiration. The procedures followed by the nodes correspond to the steps followed during the dissemination of a PCache message. Nodes will therefore put a received REGISTER on hold and process it according to the rules given by PCache; cache updates also follow the PCache algorithm rules. The goal of the SIPCache registration process is of efficiently spreading users' bindings in the MANET. Note that PCache does not aim at ensuring that each node receives a copy of each item; rather, data replication is achieved using a combined probablistic and deterministic approach, which can lead to the fact that nodes that hold a copy of a given SIP binding do not receive a refresh registration. Vice versa, nodes that previously had no knowledge of a refreshed binding can locally store in their cache the new information.

This approach on one hand, does not prevent a binding expiration in a node's cache, but on the other hand, allows for better redistribution of bindings in the network, as the number of replicas of a binding remain more or less constant within a network with given number of users. When a binding expires in a node cache, its cache entry is freed, and it is filled upon the next SIPCache message processed by that node. Similarly, if a node receives a refresh SIPCache registration message, if PCache rules dictate that the binding should be stored, this will happen replacing one of the items already present in the cache (of course, provided that it was already full).

According to PCache, the SIPCache registration messages carry as mandatory item the binding of the registering user, and in the body, the complementary items used by PCache to leverage an even distribution of replicas in the network. SIPCache complementary items are the bindings of other users in the network. An example of SIPCache REGISTER message, for the case of an original transmission, could be[5]:

```
   REGISTER sip:10.10.18.255 SIP/2.0
Via:  SIP/2.0/UDP 10.10.18.1:5082;
From:  <sip:bob@example.com>;
To:  <sip:bob@example.com>
Call-ID: 1762110646@10.10.18.1
CSeq:  0 REGISTER
Contact:  <sip:bob@10.10.18.1:5082; ver=1>
Max-Forwards:  10
Expires:  7200
Content-type:  text/PCache;
```

[5]Some SIP header fields parameters are not shown here for clarity

```
Content-Length:  1000; 15

Complementary:
AOR=sip:alice@example.com;
IP=10.10.18.45;
ver=1;
exp=3440;
sis=t;
AOR=sip:carol@example.net;
...
```

The PCache mandatory items are embedded in the SIP header fields: the From field indicates the AOR of the registering user, the Contact field reports the IP address and the version of the item being disseminated. The complementary items are reported in the body of the message: the AOR and IP address, as well as version number and expiration time are reported. The sis value, stored in sender, indicates whether the item is stored in the cache of the message sender. The message is carrying information on the user Alice, her binding and its expiration time and version number. Note that all these elements are reported on a single line, but here the line is broken for clarity reasons. A real line break indicates that information on the next item (binding of the user Carol) is now beginning. The Content-Length field indicates that the body size is 1000 bytes long and carries 15 complementary items.

5.12.3 SIPCache Query

A SIPCache query allows retrieving the IP address of a user using the AOR as a query target. An example message could be:

```
   MESSAGE sip:george@example.net SIP/2.0
Via:  SIP/2.0/UDP 10.10.18.1:5082;
From:  <sip:bob@example.com>;
To:  <sip:george@example.net>
Call-ID: 1097740201@10.10.18.1
CSeq:  0 MESSAGE
Max-Forwards:  1
Subject:  Query
Content-Type:  text/PCache
Content-Length:  1000; 15

Complementary:
...
```

The considerations presented for the REGISTER message are valid here as well. The target user is indicated in the Request-URI and in the To header field. We assume, for simplicity, that also the MESSAGE header fields are 300 bytes long, so that 15 complementary items can fit in the message body. An example response that a node may return is:

```
   SIP/2.0 200 OK
Via:  SIP/2.0/udp atlanta.com:5060;
From:  <sip:george@example.net>;
To:  <sip:bob@example.com>;
Call-ID: 1097740201@10.10.18.1
CSeq:  0 MESSAGE
Contact:  <sip:george@10.10.18.54:5043>
Expires:  4566
Content-Type:  text/PCache
Content-Length:  1000; 15

Complementary:
...
```

The Contact header field reports George's IP address and port. The message carries in the body complementary items need by PCache to leverage the geographic distribution of the bindings. Note that the value of the Call-Id header fields matches the value of the request for which the response is being returned. In this way it is easy to associate to a query the pertaining replies. The reply message can be sent by the targeted user himself (George in this case), or by any other user that has a non-expired binding for George; in this latter case, the binding of the replying user is added as one of the complementary items.

5.12.4 SIPCache Query-All

By issuing SIPCache Query-All messages a node can look for the complete bindings of users who match some specified search criteria. In general, this type of query can be used as a general service discovery mechanism, where search criteria are provided and replies on the address of the services matching the search criteria are returned. A SIPCache Query-All message looks like:

```
   MESSAGE sip:all@pcachenet.net SIP/2.0
Via:  SIP/2.0/UDP 10.10.18.1:5082;
From:  <sip:bob@example.com>;
To:  <sip:all@pcachenet.net>
Call-ID: 1097740201@10.10.18.1
```

```
CSeq:  1 MESSAGE
Max-Forwards:  8
Subject:  Query-All
Content-Type:  text/PCache
Content-Length:  1000; 5

Search:
sex=female;
int=sky,reading,music;
dur=3600;
Found:
AOR=sip:alice@example.com;
IP=10.10.18.45;
ver=1;
exp=3440;
....  Complementary:
...
```

Nodes understand that this is a Query-all message looking at the Subject header field value. This makes them ignore automatically the values of the Request-URI and of the To header field: here we placed some generally meaning value. The relevant query information is carried in the message body.

The search parameters are preceded by the ”Search:” line. Search parameters can be arranged or expressed in any possible way, implementation dependent. Here we can see that Bob is looking for a female user, who is interested in sports, reading or music, and would like to communicate for up to 1 hour. The message is being forwarded by one of the cluster heads, as defined by the PCache recollection algorithm described in the previous section. The cluster head adds to the message the information on the found users matching the search criteria; this information is located after the ”Found” marker. Note that due to the presence of the search parameters and of the information on the matching users, the message carries less complementary items.

The format of the body carried in a SIPCache message is merely an example; being SIPCache messages native SIP methods, they can carry transparently any type of body. Therefore, other methods can be used to describe the body, such as structured languages like XML or even a binary format. In order to leverage the properties of PCache, it is advisable to use a compact format, so that the number of complementary items that fit in the body of the message is maximized. Studying how the message body format (text, XML, or binary) affects the performance of SIPCache is a subject for future work.

Chapter 6

Implementation Considerations

This chapter describes the actual implementation of decentralized SIP, in terms of software architecture, and modules of which it is comprised. An example of message exchange is given, to clarify how the semantics of SIP header fields have been changed to trigger ad-hoc operations. A prototype implementation of dSIP has been realized and successfully tested in a testbed network formed by Linux laptops connected in ad-hoc mode; an evaluation of SIPCache has instead been carried out by means of simulations and will be presented in Chapter 10. Qualitative considerations on the implementation of decentralized SIP are presented to conclude the chapter.

6.1 Software Architecture

In order to support the dSIP[1] server-less functionalities, the SIP software stack of a device needs to be modified. The enhanced SIP stack must maintain interoperability of operations with SIP servers and other SIP clients, and be transparent to the applications using the baseline SIP as the signaling protocol; existing applications do not need to be modified, even though the SIP stack is extended to support operations in ad-hoc (or proximity) networks.

The software architecture of decentralized SIP is comprised of several modules, arranged in a layered structure. Fig. 6.1 depicts the structure and the connection between the modules. The main idea is that basic SIP proxy and registrar server capabilities are added to the stack of a SIP client (user agent, UA). Server and UA are further enhanced by adding extensions able to support ad-hoc operations. We have used as reference implementation, the OSIP open source software stack, together with its user agent API eXosip and the Partysip proxy server[2],

[1] In this chapter we use the abbreviation dSIP to refer to decentralized SIP as a whole

[2] Documentation and source code of OSIP and related software can be found from `http://`

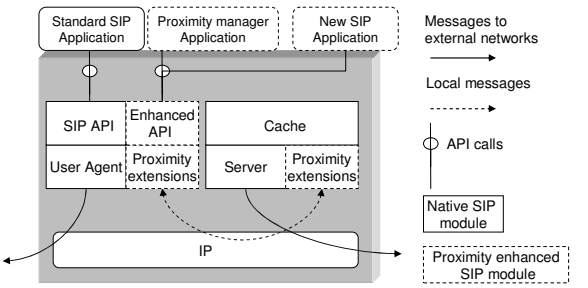


Figure 6.1: Decentralized SIP Software Architecture

which are standard SIP UA and server software components. The architecture shown in Fig. 6.1 has been implemented and evaluated in a testbed network of Linux laptops connected with WLAN in ad-hoc mode [67].

The UA itself is enhanced with capabilities for supporting ad-hoc operations. One of the modifications that the extended UA brings about consists in forwarding to the local server the REGISTER message, which would instead be sent to the predefined outbound proxy server when the node is in the Internet. When the registrar server receives a REGISTER message from the local UA it performs a sort of network registration on behalf of the UA to advertise the local user's bindings and gather the other users' contact information, storing them in its cache. The network registration is performed by the proximity extension of the server module. The local proxy server acts as a standard SIP server forwarding the requests generated by the local UA to the other nodes in the network, using the contact information gathered during the network registration phase.

Another example of proximity-specific operations is that UA can request the local server for the list of users currently available in the ad-hoc network. In proximity networks, in fact, users cannot assume, as they would do in the Internet, that it is possible to call any of their online contacts. Rather, it is very likely that users do not even know the AOR of users in the proximity. For example, the situation of travelers waiting for their flight in an airport fits in this use case. Therefore, dSIP also provides means for requesting the list of users in the ad-hoc network. Once the list is available, the user picks up one name and begins a SIP session as she would do in the Internet. The INVITE message is forced to the local proxy server (rather than to the predefined one) by the UA proximity extension and correctly forwarded as the proxy server knows the bindings of the users in the network, following the registration operations.

Local proxy and registrar servers have different functionalities, but are logically implemented as a single server. The fact that every device has locally in-

`www.gnu.org/software/osip/`, accessed October 9, 2007

tegrated SIP server capabilities, allows all the nodes to store the bindings of the other users in the network, and locally associate a SIP address into the corresponding IP address. In other words, the functionalities of registrar and proxy are distributed among all the nodes of the network.

The dashed modules, and the co-located SIP server, are our additions to a standard SIP user agent. As said, the UA and server modules are further enhanced by adding specific functionalities for deployment in proximity networks. The enhancements are realized in a modular way, as the new features extend the native functionalities, but do not substitute them. With such an approach, a native SIP application in the Internet can still use a standard SIP UA API to begin and manage SIP sessions using external proxy servers and/or be deployed in proximity networks as ad-hoc operations are triggered transparently for the application.

Another task of the enhanced UA module is to add a particular extension, .local, to the user's SIP address, when proximity operations are needed. The extension triggers proximity operations at the local server and in the remote nodes processing a SIP message containing the .local extension as Request-URI. In this way, users are still unambiguously identified, both in infrastructured and proximity networks, without needing to use two different SIP addresses.

User agent and server modules are independent, they communicate only by means of SIP messages, exchanged over UDP and sent to the loopback address. UA and server are not bound by any function call. When the device is in a proximity or ad-hoc network and the default SIP servers cannot be reached, the user agent will use the local SIP server as the predefined registrar and outbound proxy server. When the services of an external server are available, SIP messages will be sent by the user agent directly to the server as indicated in Fig. 6.1. By unpairing SIP client and server, having them communicate only through SIP messages, we enforce interoperability of our solution with server-based SIP. In fact, the local user agent can interact with the local server exactly as it would do with any external server and the application can still keep on sending registration message to the pre-configured external registrar. In proximity networks, the extensions to the UA module intercept the message and route it the the loopback address, i.e., the locally running server, rather than to the external server, transaparently for the application.

To fully make use of SIP functionalities in proximity networks, a new proximity API is introduced, and its features utilized by a proximity manager application, specifically designed for use in proximity networks. For example, a use case of the proximity manager can be to request from the local server the list of users currently present in the proximity network. Once the list has been retrieved, SIP sessions can be initiated by any standard SIP application.

We also built an application on top of the session management middleware.

The application is a simple version of an MSRP instant messaging client. The proximity manager application is used to interact with the SIP stack and to launch the instant messaging application. Using the proximity manager, the user can manage SIP sessions and registrations. Session invitation requests are received and sent by the human user through the session management application. If the session is accepted, either remotely or locally, the MSRP instant messaging application is launched with the connection parameters negotiated with SIP, i.e., IP address and TCP connection port.

The SIP modules sit on top of the IP software stack of a device. Thus, when dSIP is utilized together with PCache in multi-hop ad-hoc networks, no part of the session management software architecture needs to be modified. PCache algorithm is logically placed between the SIP middleware and the IP stack of the device. For example, when a REGISTER message is sent to the network by the local server, instead of being broacast as it would be in proximity networks, it is spread according to PCache data dissemination algorithm rules.

6.2 Prototype Implementation

Decentralized SIP has been implemented on Linux machines and the following scenarios tested on laptops connected with IEEE 802.11 wireless cards, running in ad-hoc mode:

1. Session Management with fully distributed SIP
2. Session Management with SLP-aided SIP
3. Session Management between nodes in an ad-hoc network and nodes in an external network
4. Advanced session management operations in ad-hoc networks

In this chapter, we discuss the basic session management operations, resulting from the presented decentralized SIP solution. Chapter 8 describes how a gateway allowing operations in heterogeneous environment has been implemented: the reference prototype implementation is the number 3 above. Chapter 9, which refers to the prototype implementation of the scenario in bullet 4 above, describes more advanced session management services in heterogeneous environments. These services include security certificate exchanges between ad-hoc users, request for third-party security certificates from a trusted source, and MSRP sessions between a node in the Internet and an ad-hoc node using an MSRP relay.

6.2.1 Fully Distributed SIP

For practical reasons, the tests were performed using a link-local environment. The software configuration used was the one shown in Fig. 6.1. This means that SIP was used to signal an instant messaging session, through the proximity manager application. Fig. 6.2 gives a high level representation of the messages exchanged in fully distributed SIP, in order to register to the ad-hoc network and discover the identities of users in the ad-hoc network. Section 6.3 will show a more detailed sequence of message exchange, also comprising session initiation operations.

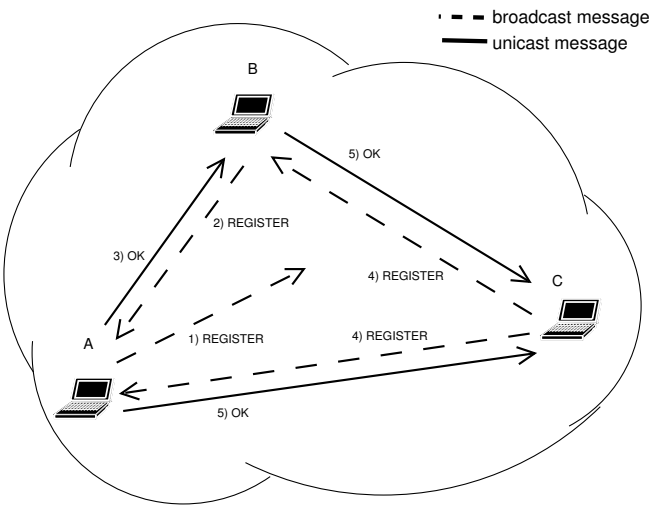


Figure 6.2: Registration with Fully Distributed SIP

During the tests, we brought the laptop A first in the ad-hoc network; node A sent a broadcast REGISTER (1), but received no reply. When node B arrived, it sent a broadcast REGISTER (2); we saw that node A received it, stored the bindings of B and sent a 200 OK containing its own bindings (3), according to the design. B gets to know the binding of A from the 200 OK message. Then a third node C joined. The operations and messages exchanged (4-5) are similar to when B appeared. After the exchange the three nodes knew about each other. Various session establishment operations were tested, and we saw that SIP sessions were successfully established, and users could engage in an MSRP chat session. The design behavior for refresh registrations and unregistrations also proved to be effective and working. For example, when an unregistration was broadcasted in the network, the other nodes server modules removed the binding for the unregistering user from their cache.

6.2.2 SLP-aided SIP

The SLP-aided SIP scenario, illustrated in Fig. 6.3, is similar. We enabled the possibility of searching with SLP for the SIP service when joining the network. The functions of broadcast REGISTER are made by the SLP Query message, while the SLP reply message returns the bindings of the available users. In the experiment, the same order of arrival than the fully distributed case was kept. Thus, at the end of the exchanges in Fig. 6.3, we saw that node C possessed the bindings of nodes A and B; node B had the bindings of A (but not of C) but node A knew no one, as expected.

Targeted queries with SLP were also tested. We made node A establish a session, indicating manually the AOR to contact, with one of the ”newer” users. The subsequent INVITE request was processed by A's local server, which looked for the list of users locally cached. Being the invited user not present in the server cache, an SLP query for the service SIP and having the invited user's AOR as attribute filter was issued. When A's server received the reply, containing the IP address, it sent the INVITE to the server module running at the address previously retrieved with SLP. A similar scenario is realizable with fully distributed SIP as well; users can invite someone who is not present in the list returned by the server. In this case, the server will send a targeted broadcast REGISTER. If no reply is received, a proper SIP response is returned to the user agent.

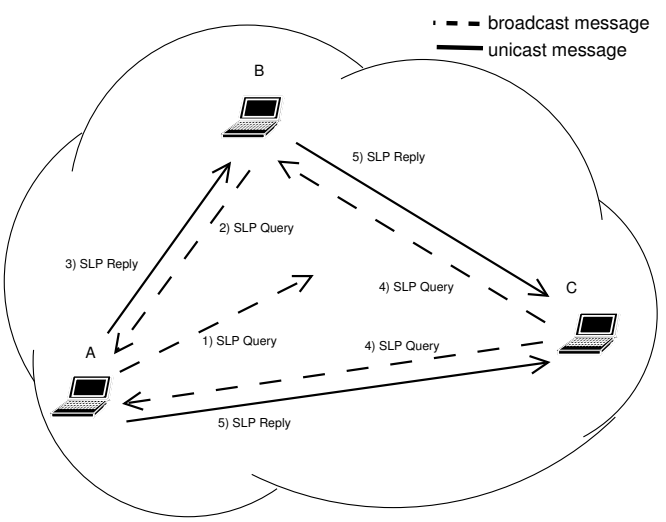


Figure 6.3: User Discovery with SLP-aided SIP

6.3 Message Exchange Example

This subsection illustrates the format of SIP messages exchanged in an ad-hoc network for a proactive dSIP exchange. The messages refer to a real test run with three laptops connected with IEEE 802.11b WLAN in link-local ad-hoc mode, that is, in a single hop ad-hoc network. The address space used in the example was 10.10.18/8.

Alice, Bob and Carol are the SIP users involved in the exchange. The messages have been retrieved from the log file written by Bob's server module. For sake of visual clearness, some header field attributes (like tag or branch), which are not relevant in this context are not shown in the message text. Their use coincides to baseline SIP specification. The SDP body is not shown as well. Fig. 6.4 shows a simplified chart of the messages exchanged in this example scenario.

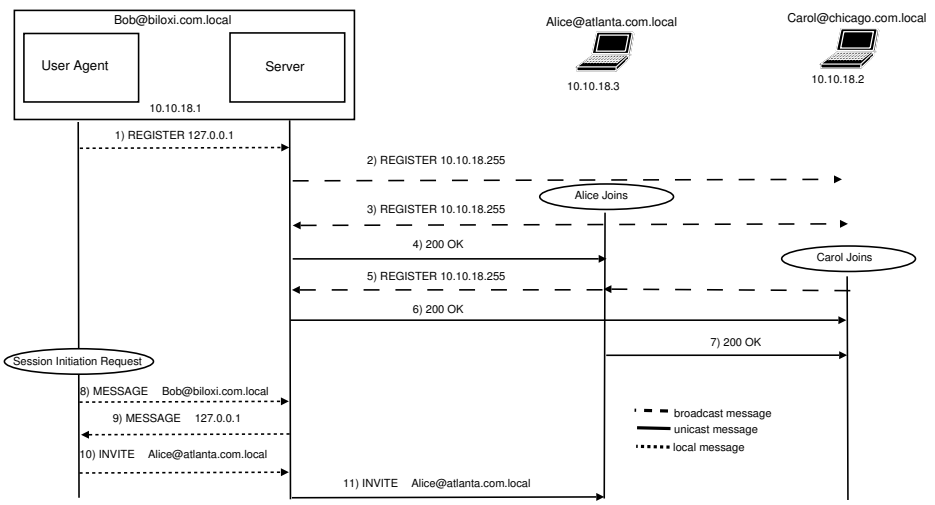


Figure 6.4: Simplified proactive dSIP message exchange chart

The chart is simplified because the provisional responses that SIP entities exchange, for some requests, are not shown, neither in the figure nor addressed in the discussion. For Bob's device only, a further level of detail is provided, indicating the message flow between the device internal modules. The flow in the other network nodes internal modules is similar. Different line styles indicate whether the message is broadcast, unicast, or exchanged internally in Bob's device, between user agent and server modules.

The first message is from Bob's user agent, which registers locally in Bob's server (message 1 in Fig. 6.4).

```
REGISTER sip:127.0.0.1 SIP/2.0
```

```
Via:  SIP/2.0/UDP 10.10.18.1:5082;
From:  <sip:bob@biloxi.com.local>;
To:  <sip:bob@biloxi.com.local>
Call-ID: 1762110646@10.10.18.1
CSeq:  1 REGISTER
Contact:  <sip:bob@10.10.18.1:5082>
Max-Forwards:  5
Expires:  7200
Content-Length:  0
```

The Request-URI in the message, *sip:127.0.0.1*, tells Bob's server module that this is a local registration. The server stores information about the local user and sends a broadcast REGISTER copying local user's contact data (2). The AOR is taken from the "From" field, while the IP address and port from the "Contact":

```
REGISTER sip:10.10.18.255 SIP/2.0
Via:  IP/2.0/udp biloxi.com.local:5060;
From:  <sip:bob@biloxi.com.local>;
To:  <sip:bob@biloxi.com.local>
Call-ID: 958890830@10.10.18.255
CSeq:  1 REGISTER
Contact:  <sip:bob@10.10.18.1:5082>
Max-Forwards:  5
Expires:  7200
Content-Length:  0
```

Every node that receives this message, stores Bob's contact information; the Request-URI *sip:10.10.18.255* triggers a reply with a SIP 200 OK message. In this example, Bob was the first user in the ad-hoc network and he does not receive any reply to the broadcast REGISTER. Later on Alice joins the ad-hoc network, sends a broadcast REGISTER (3), which reaches Bob's device. Alice's REGISTER is not shown here for brevity; its format is similar to Bob's broadcast REGISTER. The reply that Bob's server module sends (4) is:

```
SIP/2.0 200 OK
Via:  SIP/2.0/udp atlanta.com.local:5060;
From:  <sip:bob@biloxi.com.local>;
To:  <sip:Alice@atlanta.com.local>;
Call-ID: 954696999@10.10.18.255
CSeq:  1 REGISTER
Contact:  <sip:bob@10.10.18.1:5082>
Expires:  7200
Content-Length:  0
```

This message is quite different from the one that a baseline registrar would

send. The ”From” field contains the AOR of the user replying the message (in this case, Bob), instead of Alice’s AOR. Also the ”Contact” contains Bob’s IP address and port, instead of Alice’s. When Alice’s server module receives the 200 OK, it creates the binding for Bob. The ”Call-ID” of the 200 OK is the same of the REGISTER message sent by Alice. Entries in a node cache are deleted after the amount of time in seconds indicated in ”Expires” header field, 7200 seconds in this case. Later, a third user, Carol, joins the network. She receives two replies to her broadcast REGISTER (5,6,7).

Later on, Bob wants to initiate a session. His user agent requests the local server a list of users currently in the ad-hoc network by sending a MESSAGE message (8); we recall again that this message does not go through the air, but it is sent internally in Bob’s device:

```
MESSAGE sip:bob@biloxi.com.local SIP/2.0
Via:  SIP/2.0/UDP 10.10.18.1:5082;
Route:  <sip:127.0.0.1;lr>
From:  <sip:bob@biloxi.com.local>;
To:  <sip:bob@biloxi.com.local>
Call-ID: 1097740201@10.10.18.1
CSeq:  20 MESSAGE
Max-Forwards:  5
Expires:  120
Subject:  RequestUserList
Content-Type:  text/plain
Content-Length:  23

I request the user list
```

The .local extension in the Request-URI tells the server that this request must be handled according to dSIP operations. The method name, MESSAGE, tells the server to look at the ”Subject” header. The keyword in the header makes the server look into its cache and retrieve the AORs of the users on-line. In this case there is no need for the user agent to specify a body, as all the relevant information is given in the ”Subject” header field; however, a short explanatory text is added to respect the logic of the MESSAGE method. The reply that the local server sends (9) internally is:

```
MESSAGE sip:127.0.0.1 SIP/2.0
Via:  SIP/2.0/udp biloxi.com.local:5060;
From:  <sip:bob@biloxi.com.local>;
To:  <sip:bob@biloxi.com.local>
Call-ID: 448975746@127.0.0.1
CSeq:  1 MESSAGE
```

```
Max-Forwards: 5
Subject: ReturnUserList
Content-Type: text/plain
Content-Length: 59

sip:Carol@chicago.com.local
sip:Alice@atlanta.com.local
```

Of course, message 9 is logically a reply, but in practice it is a SIP request. Also, the 200 OK responses sent by user agent and local server as response to any received MESSAGE request, have not been shown and discussed, as they do not give relevant information. In this case the Request-URI is constituted by the loopback address. Using the loopback address is equivalent to add the .local extension at the end of the AOR. The subject line is set to ”ReturnUserList”, and the body of the MESSAGE contains the AORs of the users in the network. In this case the ”Expires” header field indicates the validity time of the request.

An alternative design would have been to return the reply within the 200 OK reply to the first MESSAGE. This is however a violation to the logic of operations of MESSAGE exchanges: the user data is carried in the body of the request, and the 200 OK is returned to comply with the general rules of SIP. This approach generates unnecessary signaling traffic, and this is one of the main reasons why long MESSAGE exchanges are discouraged, in favor of session-based messaging. Anyway, given that this exchange happens entirely within a node, no bandwidth is unnecessary wasted. In comparison, we underline how we have decided to embed user data, the binding, in the 200 OK reply that a node returns to a registering user, rather than an empty 200 OK; this is to avoid waste of bandwidth due to signaling.

After that the local server replies, Bob’s user agent displays the information and Bob decides to initiate a session with Alice (10):

```
INVITE sip:Alice@atlanta.com.local SIP/2.0
Via: SIP/2.0/UDP 10.10.18.1:5082;
Route: <sip:127.0.0.1;lr>
From: <sip:bob@biloxi.com.local>;
To: <sip:Alice@atlanta.com.local>
Call-ID: 1645780589@10.10.18.1
CSeq: 20 INVITE
Contact: <sip:bob@10.10.18.1:5082>
Max-Forwards: 5
Expires: 120
Content-Type: application/sdp
Content-Length: 120
```

The advantage of the software architecture adapted for use in ad-hoc networks

(shown in Fig. 6.1) is clear in this scenario, as specific methods for discovering users in the ad-hoc networks are implemented in the proximity manager application. Native SIP applications would not have means of discovering SIP users in the network, as they are not designed to perform such operations. The user should know in advance the AOR of the other users in the network to begin a session using a native SIP application. This is not always possible in ad-hoc networks.

After the user list has been returned by the local server, the operations and messages exchanged follow baseline SIP operations. As a matter of principle, Bob's server can directly send the INVITE to Alice's user agent (11), as it knows her IP address and contact port. Bob's server could have returned to the user agent the bindings instead of the AOR only; this would have allowed the INVITE to be sent directly between the two user agents. However, we designed decentralized SIP so that all the SIP messages sent in the ad-hoc network are relayed through the nodes' internal server modules. This means that the INVITE (message 10) is sent to Alice's server; main reason for this choice is to achieve more control on the exchanged SIP messages. For example, as we will describe in Chapter 9, the server modules can enforce, when requested, security by modifying and properly processing a SIP message before delivering it to the local users or sending it to the network.

Other messages of proactive dSIP scenario are not shown. A refresh REGISTER message looks like an original broadcast register; it has the same Call-ID but higher Cseq number, so all the nodes in the network can discriminate whether it is an original REGISTER or a refresh. An unregistration message is a REGISTER as shown above, with Expires set to 0, same Call-ID and higher Cseq.

6.4 The User Point of View

The human user does not need to use two different SIP implementations, or two different applications in ad-hoc and infrastructured networks. The stack of the SIP user agent is complemented by the addition of the server module for use in ad-hoc networks. In a typical usage scenario, the user starts the SIP application, but he does not need to specify the operating mode. He may be prompted to specify his wish to use ad-hoc or normal mode, but this is not necessary.

When a user decides to start a session, the specifically designed session management module checks whether the device is operating in ad-hoc mode. If so, it triggers the user agent module to send a MESSAGE, requesting the user list from the local server. The SIP application shows to the user the list of his registered contacts, to be contacted in normal mode, and the list of users in the ad-hoc network. If the user decides to initiate the session in normal mode, the local server is bypassed, and all the outgoing requests are sent to the external predefined outbound

proxy server. If the ad-hoc mode is chosen, then decentralized SIP operations are executed, and all requests are sent to the local SIP server waiting for messages at the loopback address. The process is transparent for the user because he does not need to behave differently for the two modes of operations. Of course, the user can be made aware whether the device is operating in ad-hoc mode or not.

It is not even necessary to perform user discovery; if a user is certain that a contact is available, an INVITE can be directly sent to the local server. The server does not know the IP address, so it will send a targeted REGISTER (see Subsection 5.8.3) if dSIP is used. If sSIP is used, a query for the address of the node running the SIP service and with the AOR to contact as the service attribute will be issued by the server. When a reply will be received, the INVITE can be forwarded to the callee.

6.4.1 Collisions at Link Layer

When nodes are connected together on a single link, transmission of data may be done only by one node at a time, regardless of whether the transmission is targeted to a single node, or to all nodes. Thus, when one node is sending a message, the other nodes must back off and wait for their turn, controlled by the MAC layer protocol. In dSIP, a situation where several collisions are likely to happen is when a new broadcast REGISTER is received. Immediately after, several transmissions of 200 OK messages will be attempted almost simultaneously, resulting in a high risk of collisions at the wireless MAC layer. Similar considerations apply to sSIP, where an SLP Query can trigger a storm of SLP Replies.

A typical solution for alleviating this type of problems in multiple access protocols over a shared medium is to define a random back-off interval; this randomness allows for dividing the transmissions over a longer time period. We apply this idea at SIP layer making all SIP servers wait a random time interval, an application layer back-off period, before sending the 200 OK. This divides the replies over a longer interval, and diminishes the probability of collisions as messages are transmitted at different times. The random waiting time is chosen uniformly between 0 and 5 seconds after the broadcast REGISTER has been received. This period is a reasonable time frame for receiving the replies; the length of the back off period is a compromise between lower collision probability, and faster update of the users contact information. The random back-off interval can be made dynamic, and based on the number of nodes in the network. If there are many connected devices, the interval is increased, and vice versa.

6.4.2 Message Losses

Ad-hoc networks constitute a highly dynamic environment, where nodes can join and leave arbitrarily, either by their own choosing, or due to network impairments. In case of no message losses, with the REGISTER - 200 OK handshake, dSIP makes all the nodes aware of each other's contact information with an exchange of two messages only. We recall that with sSIP this is not possible, since it is not possible to store the binding of a node issuing an SLP Query.

The situation is different if one or more of the REGISTER - 200 OK messages is lost. If the original REGISTER does not reach any user, the registering node will not receive any replies. The node can retransmit the request, or just assume that there are no other nodes in the network. Instead, if only some of the nodes in the network do not receive the original REGISTER, they will be unaware of the presence of the newcomer. The situation is fixed when a refresh REGISTER is sent, either by the newcomer or by one of the nodes that had missed the original register. Such nodes will consider any refresh REGISTER as an original one and reply accordingly.

A more troublesome situation is if a 200 OK message, sent, say, by node B as an answer to the REGISTER of node A, is lost. In this case, user B knows the identity of the registering user A, but node A will not acquire the bindings of node B. Refresh REGISTER messages from node A will not provide node A with the contact information of user B, since node B will consider these REGISTER messages as refreshes, and will not respond with the 200 OK, as defined in our scheme. This situation is resolved when node B sends its own refresh REGISTER; if node A receives it, it will store user B's binding. The loss of a refresh REGISTER message is not a major problem. The worst thing that can happen if all refresh registrations of a certain node are lost is that its binding stored in the caches of the other nodes will expire. The binding will be restored when the first refresh REGISTER is successfully received. If an unregistration is lost, then the entry in the cache will be removed when its timeout expires.

Losses of INVITE messages are handled according to SIP specification: if the local proxy server does not receive a provisional response from the contacted user within timeout expiration, it will eventually retransmit the INVITE, and only after some retransmissions, notify the user agent of the problem.

Chapter 7

Security Support for Decentralized SIP

The design of dSIP has been enhanced for providing security support, both in registration and session management phases. Security for sSIP operations is guaranteed by SLP-specific schemes, for the user discovery phase, and coincide with dSIP for the session management phase. SLP specific security mechanisms are briefly described at the end of this chapter; for more details refer to the papers [57, 56, 66].

In our scheme, the biggest security problem is in ensuring the identity of SIP users in the ad-hoc network. For example, when a 200 OK reply to a REGISTER message is returned, it is very important to ensure that the reply came from a legitimate user. In other words, it is very important to verify that the sender is really the user indicated in the From header field. Similarly, when a user invites a peer to a SIP session, it is important to verify that the session is being initiated with the correct user or that the invitation is received from the user indicated in the From header field.

We studied the SIP security mechanisms, discussed in Section 2.2, to decide which one best fits our architecture and network environment. This chapter presents such an evaluation. Afterwards, we present how we have modified and applied the chosen mechanism to be deployed in our framework and provide implementation details.

We present and illustrate the security considerations for the proximity networks scenario. However, they are also valid for the multi-hop environment and combined with SIPCache. SIPCache messages can in fact be protected for security according to the methods described in this chapter, since these methods apply directly to the SIP messages, modiyfing their headers as required. SIPCache only uses a different way for propagating the messages, based on the PCache algorithm rather than broadcast; this, SIPCache messages can be protected for security similarly to the link-local case.

7.1 Design Goals for the dSIP Security Solution

Similarly to the design of the general functionalities of dSIP, the driving goal for the dSIP security solution was to reuse existing SIP security mechanisms and eventually adapt them to the peculiarities of the ad-hoc network environment. Another design strategy was that security protection should not come at the expense of excessive and disproportionate wireless bandwidth consumption and terminal resources. The chosen security solution should introduce a minimal overhead to baseline dSIP messages and should not be too complex.

As we mentioned above, in dSIP the biggest challenge is to verify the identity of the registered users. It is infeasible to assume that only trusted users are given access to the physical wireless network; dSIP signaling should therefore be authenticated, that is, the receiver of a dSIP message should be able to verify the identity of the sender and that the message has not been tampered with. Sometimes, confidentiality is needed as well: for example, closed users would like to establish a SIP session where the entire signaling messages exchange is protected for confidentiality as well as authenticity.

In any case, protecting the messages for security adds to bandwidth and resource consumption, therefore, the security mechanisms should be applied only when necessary. For example, if dSIP is used to establish a chat session with a previously unknown person, security support may not be needed. The security solution for dSIP must support broadcast messages; public key cryptography is a technique that can be used in such a case. Instead, setting up shared secrets with all potential receiver combinations is not practically feasible, so any security solutions based on shared secrets are discarded. A new IETF working group, *Better than nothing security, BTNS*[1] has been recently chartered to address one fundamental problem of IPSec: the all or nothing principle of the need of certificates signed by a trusted entity. BTNS studies ways to deploy IPSec also when self-signed certificates only are available. Studying how the work in BTNS can contribute to improve decentralized SIP security mechanisms is a possible subject for future work. Note that within the SESSI project, the concept of IPSec using self-signed certificates has been already used and implemented, as discussed later in the next chapter. Particularly, decentralized SIP uses IPSec tunnels obtained with self-signed certificates in order to connect to the gateway node.

[1] BTNS IETF working group: `http://www.ietf.org/html.charters/btns-charter.html`, accessed October 9, 2007

7.2 Evaluation of SIP Security Mechanisms

Four methods commonly used for protecting SIP for security are HTTP digest authentication [36], TLS [25], IPsec [59], and S/MIME [96]. Additionally, the architecture for authenticated identity management in SIP [93] described in Section 2.6.4, has recently been proposed; we will refer to this method as SIP Identity.

Of these methods, HTTP digest authentication was ruled out because it relies on shared secrets. A broadcast message, or in general, a message meant to reach multiple recipients, cannot be protected with digest authentication as it would be prone to security holes; moreover, this protection scheme is unpractical. In fact, if the same secret, i.e. password, is shared among multiple users, then, as a matter of principle, authentication for broadcast messages could be possible. However, this approach leaves the door open to impersonification attacks. Any user of the group that shares the same secret can claim to be any other user of the same group. On the other hand, sending a different unicast message, with a different challenge, to all the recipients is unpractical and often impossible as usually ad-hoc nodes do not have knowledge of the IP address of the other nodes in the network.

TLS was discarded because it does not support broadcast or multicast operations, being based on TCP. IPsec, similarly to TLS, provides a lower layer hop-by-hop secure channel for SIP communication. Even though IPsec basically could be used to protect multicast messages, this requires a complex key management structure and so it is not applicable to dSIP, either.

A new security mechanism, called datagram TLS (DTLS) [84, 98], has been recently proposed to enable TLS-like protection at transport layer over a datagram protocol like UDP. If SIP would run over DTLS, confidentiality of messages would be achieved without excessive overhead, and at the same time broadcast operations would be supported. DTLS is a relatively new proposal; the specification defining the use of SIP over DTLS [48] is still work in progress, and its development to a standard status seems to be halted. Moreover, addition of dTLS support involves changes to the transport capabilities of lower levels of the SIP stack, which was out of scope for this dissertation, which focuses more on middleware issues. For such a reason we have not originally taken DTLS into consideration for integration with dSIP, but we plan to address the integration as future work.

The two remaining candidates, S/MIME and SIP Identity, are better suited for our scenarios. Both are stateless: the signature and all related information are carried in every message, and so S/MIME or SIP Identity protected messages can be broadcast or multicast. S/MIME offers integrity and, optionally, confidentiality for the message body. The message header can also be protected, but then the entire SIP message must be tunneled within an outer SIP message. S/MIME messages easily grow large due to the tunneling and the way the message is organized

into different parts. Additionally, a sender can include his certificate in the message, making the size of the message grow even more. Adding a certificate to an S/MIME body is needed when the recipient does not have the signer's certificate locally stored.

In our network environment, minimizing the overhead is a major design goal, and S/MIME does not meet such requirements of low overhead. Since SIP Identity uses remarkably less bandwidth, it was chosen for protecting authenticity in dSIP. We recall that in the SIP Identity mechanism, some sensible header fields of the SIP message, and the body, are protected for integrity by having the message signed by a trusted entity. The signature is computed by creating a hash of the header fields and the body; the signature is added to the Identity header of the SIP message. The overhead of the approach only consists of the length of the signature (if RSA-1 is used, as suggested by the specification, its length is about 170 bytes) and of the associated header fields. The receiver of the signed SIP message can verify the signature using the certificate issued by the trusted signing entity. Another advantage is that, if the certificate is not locally stored, the verifier can fetch it from the address provided in another associated SIP Identity header field. There is no need, as in S/MIME, to add a long certificate to the message. With SIP Identity, the total overhead due to the mechanism is approximately 240 bytes [93]. In comparison, protecting a message for integrity with S/MIME would have cost over 1300 bytes, due to its tunneling mechanism (see section 23.4 of the SIP specification [106]). Moreover, S/MIME is quite heavy to implement in a end-user device, possibly with limited capabilities; in fact, it is currently uncommon to find small portable devices implementing such a technology.

The biggest disadvantage of the SIP Identity scheme is that it does not protect for confidentiality. When confidentiality for the SIP messages is needed, S/MIME should be used, as it is the most feasible solution in our network environment. TLS and IPsec, which could be used to provide confidentiality at lower levels, cannot be used at a full extent in our network environment, as we motivated above, at least when messages need to be broadcast in the network. We still use IPSec when connecting to the gateway node. Due to the high overhead that S/MIME brings about, we have not chosen to further enhance the dSIP security for confidentiality. The advantage that encryption of signaling messages would bring about are not worth the consequent additional complexity and bandwidth consumption, within the bounds posed by our security mechanisms design goals. If encrypted communication is needed or desired, it should be achieved, case by case, by the application. How the application, whose media session has been signaled with SIP, enforces confidentiality is out of scope of this dissertation; as an example, our MSRP application can interact with the security service provided by the SESSI framework to establish sessions where the point-to-point instant mes-

saging exchange is done over IPSec.

However, in some cases, SIP is used at the application level, other than as the signaling protocol. This is the case of instant messaging sessions based on SIP. Also in this case, there is no strict need to use S/MIME for encryption. The signaling exchange for establishing a SIP-based instant messaging session can be done with no encryption, the actual exchange of messages (MSRP messages, see Section 2.7.2) can be done over a point-to-point TLS connection. TLS cannot be used for multiparty sessions, like chat sessions where more than two users are involved. How multiparty sessions should be done with MSRP is not yet well defined in the SIMPLE working group.

7.3 Applying SIP Identity to dSIP

We have modified the SIP Identity mechanism [93] to fit the ad-hoc networking environment. The key idea is that the message is signed with the certificate of the domain of which the signing entity is responsible. In other words, a domain server, which authenticates the message sent by a user, signs it using its domain certificate. The signature, if successfully verified, guarantees the recipient of a message that the message has not been tampered with and also that the message comes from the user indicated in the From field.

In isolated ad-hoc networks such a scheme is not feasible. There are no domain servers that ad-hoc nodes can trust to sign a SIP message and enforce the sender's identity. Rather, according to the decentralized philosophy of dSIP, we impose that each node will sign the message for itself, using a self-signed certificate. This mode of operation is allowed, although not recommended. We assume that certificates have been pre-distributed and verified previously; our scheme protects the identity of users who share a security association before they have connected to the network. For example, two work colleagues have exchanged their certificates and stored them in their devices. If they meet in a conference and want to discuss sensible issues, they can use dSIP with SIP identity enhancements to ensure the identity of the parties involved in the communication.

Another difference of our dSIP Identity scheme from the original mechanism is that SIP Identity only applies to SIP requests. Instead, we verify the sender's identity of every message by having nodes sign also SIP responses, whenever authenticity is requested. Another modification to the base mechanism is that we apply it to REGISTER requests as well: protecting the authenticity of dSIP REGISTER messages is of outmost importance in order for the recipients of a REGISTER request to verify the identity of the registering user. The SIP Identity specification does not address in detail how to apply the mechanism to REGISTER requests, leaving the issue as future work. One of the reasons why standard SIP

Identity does not address REGISTER messages is that a registrar cannot logically act as the authentication service for registration messages because it has not yet authenticated the user. This is not a problem in dSIP since each device hosts its own authentication service.

From an implementation point of view, the philosophy of SIP Identity is respected when the scheme is used in proximity networks. All the signing and verification functionalities are relied to the local server module. The user agent module never signs SIP messages. Signing is performed by the local server module, which always processes the outgoing dSIP messages sent by the local user agent and act as its authentication service. Users' certificates are not stored in the server's cache, but in a dedicated security module, which implements all the security related functions. We call this module *Authentication and Authorization (AA) Module*. The local server can interact with the AA module for retrieving the certificate of the local user and sign outgoing messages. Certificate of remote users will be fetched by the server from the AA module in order to verify an incoming signed dSIP message.

We have extended the basic authentication scheme for exchanging certificates in ad-hoc networks with previously unknown users using SIP. The starting point is the SIP Certificate Management Service [49]. The extension allows users to establish secured dSIP sessions even if they did not share a security certificate in advance. Note that if a certificate is received from a previously unknown user, the self-signed certificate should be only temporarily accepted, as its authenticity cannot be verified in an isolated ad-hoc network[2]. The Certificate Management mechanism for SIP allows also other more advanced security services. We will describe in more details these advanced security services in Chapter 9.

7.4 The Authentication and Authorization Module

The AA module has been added to our architecture, to complement the basic implementation described in Section 6.1 and enhance it with security support. This module has been realized by the Helsinki University of Technology as part of the SESSI project; its full design and the features are illustrated by Kurian [64] while the services it enables have been presented in the SESSI journal paper [56]. Such a module is also used for securing the SLP exchanges. The integration of the AA module with the decentralized SIP software architecture is shown in Fig. 7.1.

The AA Module interacts with the server and the Service Discovery (SD) module only. In principle, the AA module can be used by the user agent or even by the application directly. However, to respect the working logic of SIP identity we decided to leave the security related decisions to the server. The key idea

[2]Unless the involved users walk by each other and verify personally their identity

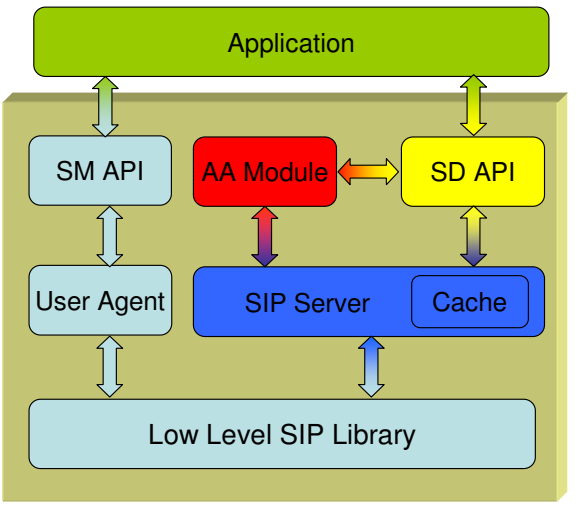


Figure 7.1: Decentralized SIP Software Architecture with AA Module

behind the AA module is that it acts as unified security module for all the services and applications running in a device. For such a reason, the service discovery service, which is independent from the session management service, can access the module as well.

To enforce security, it is necessary that a node can identify which user it currently communicates with, even though that user has different authentication credentials and user names in every service. To achieve this, the AA module uses personal hierarchical certificates. Each user has a base identity and a public/private key pair, that he uses to certify all his other service-specific keys and names. A user that wants to act in different roles can simply have several base identities with corresponding service-specific keys and names. In ad-hoc networks, in fact, for a given service, users can act as service providers or service consumers. For sake of generality, we assume that the authorization decisions are different based on the role of usage of a service. For example, user A may want to authorize user B to use the service he provides, but may not want to utilize the services provided by B. The AA module can be configured in such a way. It is also possible to define a general device security level, which overrides all the service-specific security levels to enforce an overall higher security level for all the services running in the device.

The base certificates can be self-signed or signed by a trusted certificate authority. Certificate authorities are appropriate sources of trust, e.g., when employees in the same company authenticate each other. On the other hand, users who do not know each other in advance may simply verify each other's credentials or have a friend recommend the new user. The users then rely on a distributed trust model

similar to the one used in Pretty Good Privacy [4]. The authentication and authorization component takes care of locating the correct authentication credential. It also makes access control decisions.

Additionally, the component has one further critical functionality: it allows the local user to decide what security level should be used with each user and each service. For SIP, we use two security levels: none and authenticity. However, the AA module is able to provide a higher security level for other services; for example, the service discovery service can also be protected for confidentiality [66]. Users may desire to have secure communication for particular services, and insecure communication for others. If a user tries to contact another user with an unsigned message when authenticity is required, the authentication and authorization component denies access. Services are still responsible for executing whatever authentication and confidentiality measures defined in the service; e.g., the security mechanism for dSIP is external to the AA module, but it is enforced by means of the AA module. So, it is possible that the general security level for the device is none, but that for some specific services a higher degree of security is requested. However, if the general security level for the device is authenticity, and the security level specific for a service is none, that service will anyway be used with a security level of authentication.

The AA module is used to decide what security level should be used and for finding the other user's certificate. The AA module internally maps the certificate to a base user and applies the defined access control rules, and only delivers the access control result to the service. The internal architecture of the AA module is shown in Fig. 7.2.

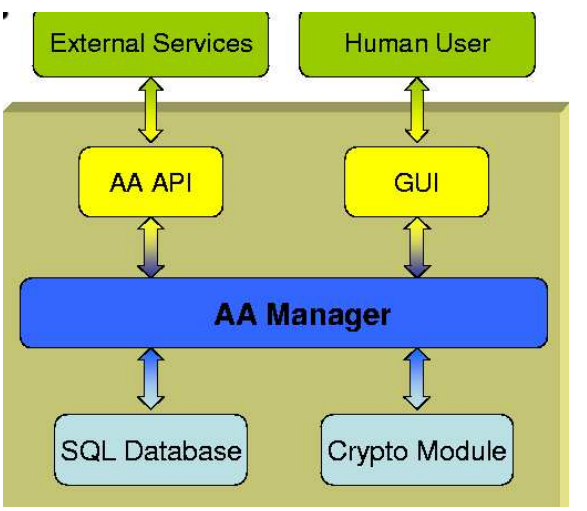


Figure 7.2: AA Module Software Architecture

The AA module is comprised of an SQL Database, where user credentials, access rules and the certificates are stored. A cryptographic module provides functions like signing and verifying message signatures. The core of the AA module is the AA manager. It enforces the security levels, manages users' credentials and handles the access control rules. The decision on whether a user is authorized or not, for which service and in which role is made by the AA manager, using the information stored in the database. A graphical user interface is provided to allow the user to change the security configuration of the AA module. For example, the user may want to change the security level for a specific service, modify the authorization rules for a specific user and so on. The AA API is provided to the services, which utilize the AA module, such as session management with SIP or service discovery with SLP. In decentralized SIP, the AA API is the module that is directly interfaced with the local server module.

7.5 dSIP Identity Usage of the AA module

When a dSIP device starts up, the server checks from the AA module (using a proper API function call) the general security level of the device. When the local server receives a local REGISTER, if a security level of authentication is used, the local server requests the AA module a signature of the relevant header fields and adds it to the broadcast (or multicast) REGISTER. Particularly, the server creates the string, concatenating all the relevant fields and passes the string, through the AA API, to the AA module. The AA module will return the signature computed over the passed string. An example broadcast REGISTER message with Identity header fields is:

```
   REGISTER sip:192.168.207.255 SIP/2.0
Via:  SIP/2.0/udp atlanta.example.com.local:5060;
From:  <sip:alice@atlanta.example.com.local>;
To:  <sip:alice@atlanta.example.com.local>
Call-ID: 879300639@192.168.207.255
CSeq:  1 REGISTER
Contact:  <sip:alice@192.168.207.2>
Max-Forwards:  5
Expires:  7200
Date:  Mon, 25 Jul 2005 10:35:48 GMT
Identity:  Q6rHlAyg/3bmXkX9cOj7BgCigYa4ZN0dvVr1i
Identity-Info:  127.0.0.1
Content-Length:  0
```

Compared to the broadcast REGISTER message shown in Section 6.3, there

are three new header fields. The Identity header field carries the signature that the local server has received from the AA module. In the example, the signature is cut out to few bytes for clarity reasons. The Identity-Info header field contains the address where the receiver can retrieve the certificate needed to verify the signature. In our case it is the loopback address, to signify both that the message is self-signed and that the receiver should have the certificate locally stored. The Date header field is optional in standard SIP, but in the SIP Identity mechanism must be mandatorily added and signed to increase the security of the digest string.

When another node in the network receives a signed REGISTER, it replies with a signed 200 OK message, even though its local security level is set to no security. This is because the registering node, having signed the broadcast message, is requesting that all the users that reply to the message are able to identify themselves. If a node has no certificate locally stored for the registering user, it will just ignore the message, and will not add the binding of the registering user to the cache. An example of a 200 OK message sent in reply to the REGISTER is:

```
   SIP/2.0 200 OK
Via:  SIP/2.0/udp atlanta.example.com.local:5060;
From:  <sip:bob@biloxi.example.org.local>;
To:  <sip:alice@atlanta.example.com.local>
Call-ID: 879300639@192.168.207.255
CSeq:  1 REGISTER
Contact:  <sip:bob@128.214.10.226>
Date:  Mon, 25 Jul 2005 10:35:23 GMT
Identity:  D2PC19FeUc3nn1DvUH2VcLIcLC7n
Identity-Info:  127.0.0.1
Content-Length:  0
```

If a node that has a security level of none registers to the network, and a receiving node has a security level of authentication, two situations are possible. If the certificate of the registering user, as indicated in the From header field, is not stored in the AA module of the receiving node, the message will be discarded. If the certificate is present, the receiving node can ”challenge” the registering user to prove his identity by sending a unicast REGISTER message. The address of the registering node is known to the replying node because of the original, unsigned, broadcast REGISTER. If a signed 200 OK reply is received from the registering node, and signature verification succeeds, the replying node can add the registering node to the list of users in the network, as the user's identity is proved. Note that the SIP Identity specification defines a proper response code, 428 Use Identity Header, to be used in such a situation. We have chosen to reply with a unicast REGISTER for a matter of logic, because the unicast REGISTER

will also serve to store the binding of the replying node, besides triggering a signed 200 OK reply.

The 428 Use Identity Header response is instead returned if a user, whose certificate is present in the callee's AA module, sends an INVITE to a peer. In this case the 428 response will trigger the inviting user to re-send a signed INVITE. If an unsigned INVITE message is received from a user whose certificate is not present in the local AA module, and security is requested at the callee's device, a SIP 403 Forbidden reply is returned and session establishment fails. The same response code is returned also if a user's certificate is found, but the user is not authorized. In general, if a server module that receives a signed INVITE successfully verifies the signature, it removes it and forwards the INVITE to the local user agent. If verification fails, or a signature is locally requested, the local user agent is not notified that an incoming INVITE was received by the server.

7.6 The Two Phases of Security

The operations of dSIP are clearly separated into two distinct phases, user discovery or registration and session management. The security support can also be considered split into two phases. For example, a user may want to discover the identities of all the other users in an ad-hoc network, and let all the other users discover his presence in the network. This is the typical behavior of open users, as defined in Section 5.5. However, open users, once they discover the identity of known contacts in the ad-hoc network, may desire to initiate secured sessions with them. The flexibility guaranteed by the AA module, by means of the functions implemented in the AA Manager module, lets the user specify the security level for each phase. Of course, the assumption that the general device security level overrides lower security levels still holds in this case, i.e., it would not be possible to behave as an open user if the general security level of the device is set to authenticity.

There is also another advantage related to the distinction of security protection mechanisms into two. When SLP-aided SIP is used, user discovery is performed with SLP. If security is needed for this phase, our architecture is not bound to use the SIP Identity mechanism, which cannot be applied to SLP. Instead, SLP specific security mechanisms [57, 56, 66] are used. The SIP server does not need to be aware of these mechanisms, nor the user. The user specifies a security level, and if authenticity is used, SLP messages will be signed and protected. With SLP there is also the possibility to use a third security level, confidentiality. If confidentiality is used, only users who have a previous security association can know who is effecting the SLP Query and can reply accordingly. Session management operations are secured with SIP Identity, independently from the fact that for the

user discovery phase SLP or SIP was used.

7.7 SLP Security Support

In this section we briefly introduce the security mechanisms used by the SLP version modified for use in ad-hoc networks. Chapter 10 will provide a comparison of the security overhead needed for protecting dSIP and sSIP; for such a reason, we provide a short summary of the security mechanisms used for SLP.

In standard SLP, it is possible to authenticate the entity that originally created a URL or an attribute list of services. In the standard SLP architecture, an entity called Server Agent (SA) can sign the URL and the lists of the services it advertises. When sending these elements to SLP clients (User Agents), an SA adds the corresponding signatures to the end of the message. The authentication related information in SLP is carried in a so-called authentication block. The basic authentication block of SLP is illustrated in Fig. 7.3. This block is added to an SLP message to provide message integrity and ensure sender's identity. For more details on the meaning of its fields refer to Section 9.2 of the SLP specification [41].

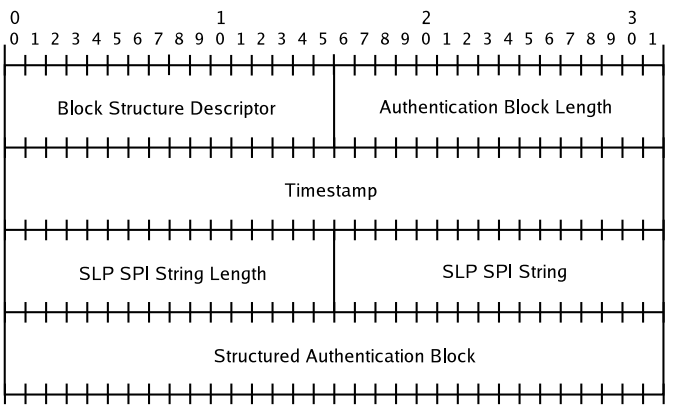


Figure 7.3: SLP authentication block ([41])

This basic structure of an SLP authentication block has been modified for providing more advanced security support, as shown in Fig. 7.4. The main problem in the basic structure of the SLP authentication block was that only the entity providing a service could be authenticated. Moreover, basic SLP does not offer means for protecting the integrity of the most common messages. Therefore, it is possible, e.g., to reuse signed information in faked messages. In ad-hoc networks, there is the need for authenticating the clients as well and protecting all the messages exchanged. The reason for this is that all the nodes in ad-hoc networks are peers, and there are no trusted centralized entities providing service information.

For more details on the meaning of the newly added header fields, refer to the SESSI project paper [56]. The enhanced authentication block is bigger than a standard one, and therefore introduces a higher overhead. In Section 10.4 we will compare the overhead due to dSIP identity with the overhead brought about by SLP with the enhanced SLP block, when performing secured user discovery.

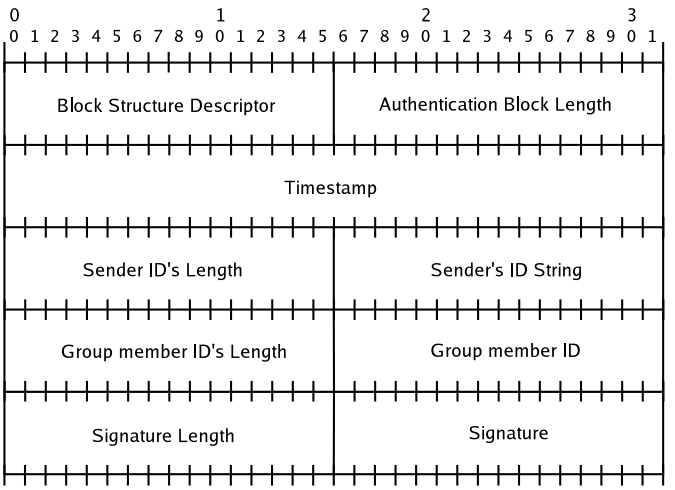


Figure 7.4: Improved SLP authentication block ([66])

7.8 Security Support in SIPCache

SIPCache can transparently reuse the security mechanisms described above. The modifications to SIP Identity described so far have been referring, for clarity, to decentralized SIP applied to proximity networks. However, SIPCache is a routing-level modification to decentralized SIP, and affects how SIP messages are spread in the ad-hoc network or the caching policies at each node. Any SIPCache message can therefore be signed according to the procedures described above, if the security level of a node requests processing secured messages.

The direct implication of signing a SIPCache message using the SIP Identity mechanism is the increase of the message headers. Bigger headers, given that decentralized SIP messages in ad-hoc networks are sent over UDP and thus their size is limited by the specification to 1300 bytes, imply that less complementary items fit in the body of the message. This lowers the performance of SIPCache, as the simulation results presented in Section 10.6.5 will show. In such a scenario, S/MIME would be very detrimental to SIPCache performance, due to the size that S/MIME bodies can reach, and this was another reason for ruling out S/MIME in

our security solution.

A different issue is whether users should trust the information carried in SIPCache messages. When the dissemination algorithm dictates that a received message must be retransmitted, the caching policies of SIPCache may impose that some of the complementary items carried in the body are modified, in order to leverage a better distribution of the bindings in the network. Some users may find unacceptable that messages are modified by intermediary entities. We identify these users as the closed users, described in Section 5.5; closed users do not modify either the content of a SIPCache message, but they simply forward it when the dissemination algorithm dictates to do so. On the contrary, open users do not care whether messages are modified by unknown parties, as their target is to know the highest possible number of people, and they facilitate the spreading of bindings in the network.

Two cases are possible when a closed user receives a SIPCache message: either the message is signed with SIP Identity or not. In the first case, if the user that signed and sent (or simply forwarded) the message is trusted by the closed user, the message content will be processed and cached according to SIPCache caching policies. If the message needs to be forwarded and its body changed, i.e, some of the complementary items are reshuffled according to the algorithm, a new value for the Identity header field will be calculated. If a closed user receives an unsigned message, the same reasoning applies: the message content can be discarded and the message forwarded according to the algorithm, adding a signature as requested by the node security level.

Note that when the content of a SIP body is modified and signed again, as in the above scenario, any node that verifies the message integrity must use the certificate of the user that has last modified the message. The assumption that the certificate to be used for verification is always the sender's one can be valid only in link-local proximity networks. However, there is a solution for this, and it is given by the Identity-Info header field, which is used to report the address where it is possible to retrieve the certificate with which a message has been signed. So, if a node modifies the body due to SIPCache operations and resigns the message, it will also add its IP address in the Identity-Info header field, substituting the previous value. The original sender always adds the loopback address 127.0.0.1 to the Identity-Info header field, to indicate that the message has been signed by the originator. Nodes can retrieve the certificate if needed from the IP address reported in the contact header field. The decentralized SIP session management framework offers means to do so, as we will describe later in chapter 10. This distinction is important as closed users may not want to store the contents of a message, although signed by a trusted user, unless the trusted user is the sender. An untrusted user could have in fact previously modified the content of a message

sent and originally signed by a trusted user. The described behavior implies that a chain of trust is established among all the nodes that forward the message, at least if all the nodes are operated by closed users. Studying ways to break this chain of trust will be subject to future work.

7.9 DoS Attack Considerations

Denial of Service (DoS) attackes are possible in every network environment, and therefore also in the decentralized SIP operating environment. This section briefly discusses, without aiming at being exhaustive, some of the possible DoS attacks that can be carried out exploiting the architecture of decentralized SIP. For DoS attacks referring to the nature of the ad-hoc networking environment, we refer the reader to dedicated literature [140].

The most straightforward DoS attack in proximity networks is continuosly flooding the network with bogus registration requests on behalf of an unaware user. This behavior may result in a DoS attack due to the flood of 200 OK replies received by the unaware registering user, and is particularly serious when the network is mostly formed by open users. In general, it is possible to enforce rules so that registration messages received from a particular IP address or AOR are automatically discarded. Unfortunately, little can be done at middleware layer to prevent the network interface from receiving a storm of bogus registration messages.

The situation in multi-hop ad-hoc networks and when SIPCache is used is more complex. Here the DoS attack can come not only from a flood of messages from malicious nodes, but also by reporting as complementary items uncorrect bindings. If false contact information is stored, on the long run it may be very difficolt if not impossible to contact legitimate users. This behavior is affected by the number of malicious nodes disseminating false information and also by their activity rates, i.e., by their rate of spreading false data into the network. The nature of SIPCache, which disseminates and replicates the advertised data items, amplifies the behavior.

A partial solution to this problem is to behave like closed users, and request all the SIPCache messages to be signed, and accept and store only those signed by trusted nodes. However, this approach may not be applicable in scenarios involving dynamic collaboration of users that join an ad-hoc network like a one-time event, such as in the case of the airport. In such situations, it is very likely that most if none of the users have a security association shared in advance. It is a suitable subject of future work to study how injecting in the network false information affects the operations of SIPCache, varying the number and the activity rate of the malicious nodes.

Chapter 8

A Gateway for SIP Signaling

This chapter presents the contribution of the second year of the project that created the research basis for this dissertation. The results have been presented in two publications [77, 68], while another journal article is in review at the time of writing.

We have so far mainly addressed the problem of communication in proximity environments, or more in general, in environments where the support of centralized servers is not available, such as small private networks. The next step towards a truly ubiquitous communication paradigm is allowing users in the Internet to communicate with users in the ad-hoc network. The assumption is that one (or more) of the ad-hoc nodes has access to the Internet itself. This can be the case, e.g., of a device equipped with a WLAN card and capable of 3G connectivity, which can use the WLAN in ad-hoc mode to connect to the proximity network while accessing to the Internet trough the cellular network. Alternatively, one could think of a hybrid scenario, where one of the ad-hoc nodes is actually a fixed node with two interfaces: one wireless to the ad-hoc network and one wired to the Internet. Such a scenario could cover wireless hot-spots, where nodes can connect with each other in ad-hoc mode (e.g., in an airport waiting lounge) but still an operator managed node provides Internet access service to subscribed users through ad-hoc connectivity.

The two above described examples imply that one of the ad-hoc nodes offers the Internet access service to the others, and possibly enforces an access control policy to permit the use only to a subset of users. This chapter deals on how to enable SIP signaling between ad-hoc networks and the Internet. However, this service is part of a more comprehensive framework, which enables ad-hoc nodes to discover a gateway node offering internet access service and implements security policies for selected and secured access to the gateway node.

The main issue when enabling communication between ad-hoc nodes and the Internet is that the ad-hoc addressing space may not be public. If ad-hoc nodes

had globally routable IP addresses, then any node in the Internet could easily communicate with them, provided that a gateway is available. When, and we believe that this is the most common case, the ad-hoc addresses are not public, it is necessary to devise a mechanism for permitting communication. This chapter describes such a mechanism, especially applied to SIP signaling operations.

8.1 Problem Statement

In order to achieve Internet-wide SIP-based communication, the ad-hoc address space should be formed by globally routable IP addresses. Non routable addresses hinder the communication between the external proxy/registrar or user agent (UA) and the ad-hoc node[1]. The problem is that the insider SIP client registers its ad-hoc IP address and communicates to the remote outsider UA the ad-hoc IP address, during the offer/answer exchange. Registrar and outsider UA would therefore, according to SIP procedures, try to send all the SIP messages to the insider at the non-routable ad-hoc IP address, which is not possible unless a route for that non-routable address is known in advance. This problem was analyzed in a preliminary paper, which presented the framework for internal ad-hoc operations and a proof of concepts demonstration on how interoperability with the Internet could be achieved [67].

To tackle this problem, we have designed, implemented and tested a SIP gateway, which runs on the ad-hoc node providing connectivity to the internet. The SIP gateway has the task to hide from the external Internet the fact that the insider does not have a globally routable address. Rather, all the external nodes would communicate with the insider through the gateway node, sending their messages to the public interface of the gateway, which will then forward them to the correct recipient.

The main design goal for the SIP gateway was to provide the described functionality respecting SIP standard procedures, in a way that a user accessing from the Internet is not aware that the remote peer is within an ad-hoc network. A minimal set of extensions to the base SIP protocols should be used, to ensure that all the involved SIP entities can support this interworking scenario. The implementation was targeted to be as simple as possible, so that the gateway service could also be run in nodes that are not extremely powerful.

By deploying a SIP gateway in one of the ad-hoc nodes, it is possible for an insider node to register to an external registrar, and establish SIP sessions, in both directions, between an insider and an outsider node. The gateway enables any kind of SIP signaling message. In this dissertation, we discuss scenarios of

[1] In the rest of this dissertation, we will refer to a node or user in the ad-hoc network as insider, and to a node or user in the Internet as outsider

insider registration to an external registrar and session establishment between an insider and an outsider. For example, presence communication between an insider and an outsider using SUBSCRIBE and NOTIFY would as well be possible, but it is not discussed here.

All these scenarios assume that the insider has knowledge of the gateway ad-hoc IP address; the address is discovered making use of the service discovery framework implemented by Tampere University of Technology as part of the SESSI project, and which we have utilized also for discovering the bindings of SIP users alternatively to using SIP-only messages. Thus, we have used a modified version of the Service Location Protocol (SLP) [41], adapted for use in ad-hoc networks [56], to allow nodes to discover the gateway address, and the gateway to advertise its service, in a distributed fashion, in the ad-hoc network. If a SIP message must be sent to the external network, it is always sent to the gateway, which will then process it and forward it to the next hop.

Communication with the gateway has been made secure by means of the security services provided within the third branch of the SESSI project, at the Helsinki University of Technology, which we also widely use in several other aspects of our decentralized session management framework. Using the security services allows communicating with the gateway over a secure IPSec session or, at the gateway side, to enforce service access to a subset of authorized users. This would be the case of an operator provided hotspot, which would provide access only to authorized customers, identified by their security certificates. We will discuss later in this chapter a bit more in detail how service discovery and security services are used by the session management framework. More details on the implementation of security and service discovery capabilities are out of scope; an article collecting the main outcome of the second part of the SESSI project, which describes the implementation and interoperability of the three services, is under review at the time of writing this dissertation.

8.2 Registration to an External Registrar

As explained above, an insider UA cannot plainly register its non-routable IP address to an external registrar. When a user in the ad-hoc network decides to register to an external server, the SIP application, unaware of the fact that the user is in an ad-hoc network, sends the message to the predefined registrar address. The decentralized SIP middleware, namely the proximity extensions to the user agent module shown in Fig. 6.1, intercepts the message and forces its destination to be the gateway.

There are several ways how the proximity extensions UA module can understand whether to force routing the message towards the external registrar. If an

application that is aware of proximity extensions is used, it is possible to build in the application the awareness of being in an ad-hoc network and offer the end user the possibility of explicitly registering to an external network. The enhanced API maps this request so that the message is routed through the gateway to the external registrar. A native application can also register to an external network transparently: when the application executes a registration request through the native SIP API, the proximity extensions to the UA module, aware that the device is in a private network, will trigger a double registration, to the local server for ad-hoc operations and to the pre-defined external registrar.

The simplified flow of messages for a registration scenario, where some messages, e.g., authentication messages, have been omitted for clarity, is shown in Fig. 8.1. The gateway receives the REGISTER message (message R1), containing the mapping between the SIP address of the registering user and the ad-hoc IP address. The gateway acts as a registrar server itself storing the binding in its local cache, and then forwards the message to the registrar indicated in the REGISTER Request-URI.

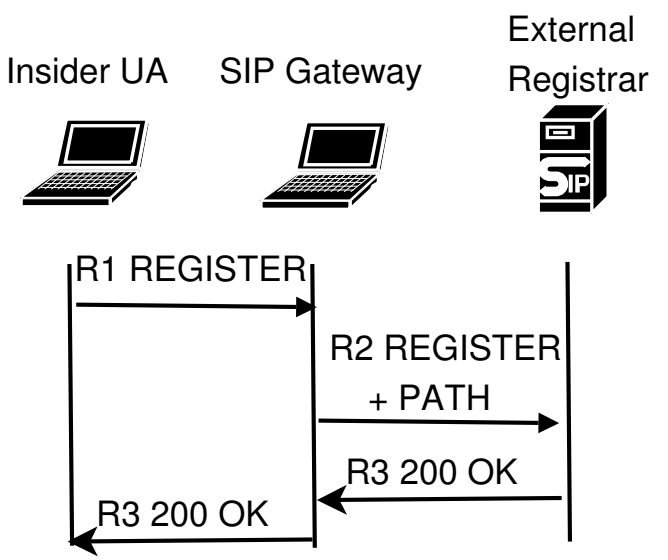


Figure 8.1: Simplified gateway-assisted registration

The forwarded message (R2) contains a new SIP extension, the ”Path” header field [138], which allows SIP registrations for ”non-adjacent” contacts. This extension is used when there is one or more intermediary SIP entities between a SIP UA and its registrar, as in the case of 3G networks. The value of this header field is the public IP address of the gateway. This extension tells to the external registrar that all the requests addressed to the insider user must be routed first through

the address present in the Path header field. After the external registrar has stored the binding with Path, the response follows the request inverse path (R3, R4), according to SIP specifications.

This approach is totally transparent for an insider client, which is not involved in the operations triggered by this extension. However, the external registrar needs to support it, and this is the only modification from the standard protocol [106] required for any external entity involved in the process. Such an extension is well known, so it is very likely that registrars support it. Even if there is no support, a workaround consists in having the gateway replace the contact address indicated by the insider, with its public IP address. Upon receiving a request actually addressed for the insider node, the gateway will substitute its public IP address with the insider's IP address and forward the message into the network. This option has the disadvantage that personal information is modified by a third party entity but can be used in case the insider's registrar signals to the gateway lack of support for the Path extension.

8.3 SIP Signaling in Heterogeneous Networks

Any type of SIP signaling between ad-hoc networks and the Internet can be enabled using the gateway. We describe here how a session can be established, but steps are conceptually similar for other types of requests, e.g., SUBSCRIBE and NOTIFY, as well.

8.3.1 Incoming Session Invitation

This section describes how an outsider user can successfully invite to a SIP session an insider user, who does not have a routable IP address, with the precondition that the insider has previously registered to its external registrar, as defined in the previous section.

The simplified scenario for incoming session invitation is shown in Fig. 8.2. An outsider A sends an INVITE to a user B, who has previously registered to her predefined registrar. The message follows the normal chain of SIP proxies until it reaches the registrar of B (message M1). The registrar retrieves the bindings for the user B specified in the Target-URI; the information registered for the insider user B comprises also the Path header field added by the gateway. The registrar, thus, does not forward the INVITE directly to the UA, at the address specified into the Contact header field, but to the address specified in the Path, that is, the gateway (M2)[2].

[2]For simplicity, we consider a scenario where the external registrar also acts as a proxy.

The gateway, acting as the registrar server for insider nodes, retrieves from its cache the ad-hoc IP address associated with the target user and forwards the message to the destination adding the Record-Route header field [106], carrying its public IP address as a value (M3). This header field tells that all the subsequent SIP requests initiated within the dialog shall traverse the gateway. The reply from the insider node follows the inverse path of the request, according to standard SIP (messages M4-6), and carries the information that the gateway has requested to stay in the SIP signaling path.

The SIP session establishment three-way handshakes concludes with the ACK request sent by the inviting user A. Normally, this request would directly be addressed to the remote peer, without passing through any intermediate SIP server. In this scenario, this is not possible as the inviting UA cannot route the ACK to the non-routable address learned during the session establishment process. However, with the Record-Route, the outsider UA does not need to try to send the message to the remote peer, but rather to the gateway (M7). From the gateway, the ACK is delivered to the insider (M8), and the SIP session is successfully established. Similarly, the session will be torn down using a BYE request routed through the gateway.

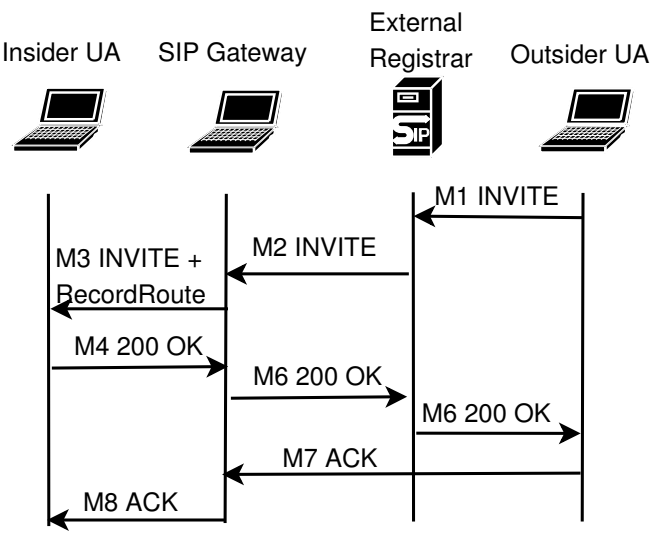


Figure 8.2: Simplified gateway-assisted incoming invitation

8.3.2 Outgoing Session Invitation

The scenario for an outgoing session invitation, where an insider invites a node in the Internet, is similar. When an INVITE is sent to an outsider (as determined by

the Request-URI, which does not contain ad-hoc specific extensions) the message is forced by the proximity extension to the network gateway. This is message M1 in Fig. 8.3.

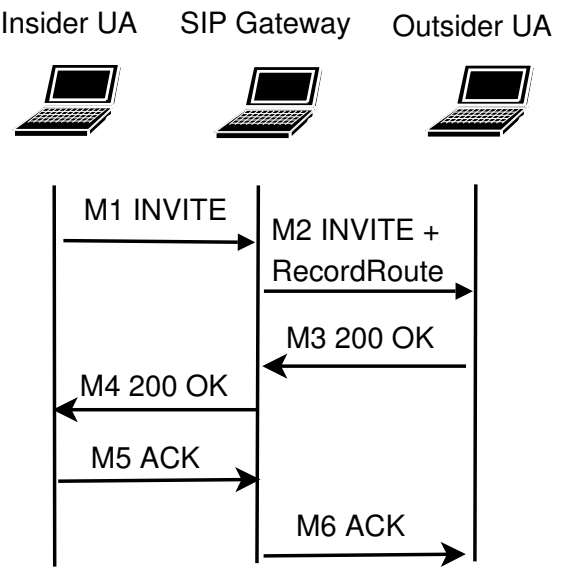


Figure 8.3: Simplified gateway-assisted outgoing invitation

The gateway adds the Record-Route with its public IP address as value and the message is routed through the chain of SIP proxies until it arrives at the invited user (M2). The response follows the inverse chain of proxies until it reaches the inviting user, according to SIP specifications. In this scenario, the Record-Route is necessary not for routing the ACK request, but to cope with the possibility that the BYE is sent by the outsider. In such a case, if the gateway had not added the Record-Route, it would not be possible for the outsider to send the message to the insider. In any case, we prefer to keep the gateway on the path of any session, to allow more control on active SIP sessions.

8.4 Interactions with the Other Framework Services

This section provides brief qualitative considerations showing how the SIP gateway access interoperates with the other two components of the SESSI service framework, service discovery and security. As part of the framework, a gateway node providing a generic Internet access service, enhanced with SLP-specific extensions has been implemented. Moreover, security support has been added as explained below. The SIP gateway is another facet of the same service. The SIP

gateway is a stand-alone application, and it can run on a node of its own, or it can be embedded in a node providing other gateway services, such as SLP gateway, as well.

8.4.1 Discovery of the Gateway

The description of gateway operations so far has assumed that the insider node knows the address of the gateway; in ad-hoc networks, which are dynamically formed, it is not possible to assume that nodes use a pre-configured gateway address. In order to find out the gateway address, two distinct routing layer mechanisms can be identified, to tie the discovery to the ad-hoc routing protocol, or to flood router advertisements into the ad-hoc network and let the nodes choose the best gateway to use.

Instead, we have chosen to perform gateway discovery at middleware layer, utilizing the services of the service discovery framework for ad-hoc networks developed within the SESSI project by Tampere University of Technology [56]. There are two basic ways to discover the gateway address using the framework.

Active service discovery is a pull type mechanism, and is used by a node if it requires to discover the gateway address: an SLP request is issued for the node providing the gateway service, and a reply containing the IP address of the node (s) providing the service is returned. A novel extension to this scheme consists in passive service discovery, where the gateway node itself advertises its service spreading into the network SLP packets containing the address of its service. Nodes interested in receiving service advertisements just listen to the incoming messages and store the gateway address for possible future use. The extensions for using SLP as service discovery protocol within the session management modules have been implemented within the extended proxy server modules. Passive service discovery is not a standard version of SLP, but it has been implemented within the scope of the SESSI project.

8.4.2 Security Considerations

Interactions with the gateway can also be enhanced by means of the third service type enabled by the SESSI project: security support. Secure interactions with the gateway are guaranteed by enforcing access control policies, so that only authorized users can connect to it and by securing communication with insiders by means of establishment of IPSec tunnels.

Both of these schemes have been logically implemented below the middleware layer. Our session management framework can therefore transparently benefit from them. It is important to define access control policies for the gateway, as it may be a small device, with few available (battery and/or computing power)

resources, willing to provide access only to a specific subset of users. Another scenario where access control is useful is for the case of an operator provided gateway, which offers toll-based Internet access to authorized ad-hoc nodes, for example, those that are customers of that operator.

On top (or instead of) of the low layer security mechanisms, we can reuse our session management specific security extensions to provide access control to the gateway, especially in the case where the SIP gateway runs stand-alone. Any SIP message forwarded through the gateway, if security is requested, is signed according to the method described in Section 7.5. If verification of the signature succeeds, the gateway forwards the message to the next SIP hop, otherwise it returns an error message to the insider node.

Chapter 9

Instant Messaging and Presence with dSIP

This chapter presents advanced services and application that we have built on top of the session management framework described so far. These services can be categorized into three main branches: instant messaging and presence (IMP) services, advanced security services and interoperability of IMP services with the Internet. By building these services on top of the middleware signaling platform we realize a complete and functional session management framework, which is able to interoperate with the Internet and provide communication means to users who do not have direct access to the Internet. On top of this, the security mechanisms provided add to the functionalities of the framework.

9.1 Presence Services in Ad-Hoc networks

The presence framework for SIP [99] can be easily adapted for use in ad-hoc networks. The specification in fact, allows that the Presence Agent (PA), the logical entity managing the presence state of a presence client, or presentity, is co-located with the presentity itself. In order to show the design effectiveness of dSIP, we have implemented a proximity manager application used for managing SIP sessions and registrations in ad-hoc and heterogeneous networks and exchange presence information with other users in the network.

The entity acting as PA of each ad-hoc node is the local server; the proximity manager application offers to the user the possibility to define a personal profile (indicating interests, age, sex and so on) and broadcast such information into the ad-hoc network using a SUBSCRIBE message. Similarly to what happens for a REGISTER message, the SUBSCRIBE is first sent from the proximity manager application through the UA modules to the local server, and from it broadcast to

the network.

We limited the presence functionalities of the proximity application to the possibility of sending a SUBSCRIBE and receiving a NOTIFY with similar presence information from other users in the network, to show as proof of concept how such a SIP service could be used in ad-hoc networks; presence-related messages were also protected for integrity applying the SIP Identity mechanism to SUBSCRIBE and NOTIFY requests and their replies in a way similar to the REGISTER and INVITE cases.

There are several extensions to the basic concept that can contibute to a richer user experience. The user can specify the profile of users whose presence information he is interested to and communicate it through the proximity manager to the local server (by means of MESSAGE requests using proper Subject). Whenever the local server receives a broadcast SUBSCRIBE it uses the received information as a filter to decide whether to forward or not the SUBSCRIBE to the user agent (this would mean for the human user a notification like *a person matching your interests has joined the network!*). The server will also send a unicast NOTIFY to the remote user whose profile is matching the local preferences.

Presence services can also be used to implement native statutes for ad-hoc users, such as active, busy or invisible, or to communicate capabilities to the other ad-hoc nodes. For example, a device may advertise its capabilities to establish instant messaging, VoIP and video call sessions, or even that it can act as a gaming server for users in the proximity.

In order for any type of device to understand the format of the exchanged presence information, a standard way to describe presence information must be used. For our proof of concept demo, we have used a simple text-based key=value format. The specification mandates to use the XML-based Presence Information Data Format (PIDF) [124] to describe presence documents within the SIP Presence Event framework. However, XML format is quite verbose and could make presence documents very heavy to carry as the body of SIP SUBSCRIBE and NOTIFY messages. In wireless networks, this is a problem to be carefully addressed. Our simple text-based presence documents are shorter than PIDF documents, but they are not standard. A suitable solution would probably be to carry the presence documents in a compressed format, after defining a standard compression method for PIDF documents.

9.2 The Message Session Relay Protocol

We have described the Message Session Relay Protocol (MSRP) [26] in Section 2.7.2. The MSRP is a protocol for session-based instant messages exchanges. In order for an MSRP session to be successfully established, it is necessary a

rendez-vouz protocol that negotiates the session parameters beforehand; SIP is a natural choice for the purpose, although any signaling protocol can be used. We have implemented a simple version of MSRP and deployed it in an ad-hoc network, as an example of application that can be used on top of the dSIP session management framework. In this section, we describe implementation details and how our framework can be used to launch MSRP sessions.

Setting up an MSRP session using dSIP in an ad-hoc network follows the standard SIP and MSRP procedures. The user selects one of the persons available in the ad-hoc network from the list that the proximity manager has retrieved from the server. An INVITE message is sent to the local server and from here to the selected user in the ad-hoc network. The INVITE message contains in its SDP body the information needed to establish an MSRP session, that is, the MSRP URI of the inviting user and the IP address where the MSRP session should be established. An excerpt of the SDP body would be:

```
   c=IN IP4 192.168.1.34
m=message 1234 TCP/MSRP *
a=accept-types:  message/cpim text/plain text/html
a=path:msrp://192.168.1.34:1234/agic456;tcp
```

The meaning of the c= line is standard: it indicates the address where MSRP sessions would be accepted. The m= line indicates the type of media exchanged in the session, instant messages using MSRP protocol over TCP on the port 1234. The first media attribute line a= indicates the content types that the node is capable of processing, while the second carries the MSRP URI.

The reply from the remote user contains the remote MSRP URI and address. Once these data have been exchanged, the proximity manager applications launch the MSRP client: the inviting user launches it after the 200 OK is received, while the invited party after the ACK is received. A TCP connection is established between the two parties and direct peer-to-peer MSRP messaging can begin. If encryption of is needed, the message exchange can happen over TLS; it is up to the MSRP application to request such a secured session. We have described in Chapter 7 the reasons why dSIP messages are not encrypted.

9.3 Relay Operations

The steps needed to establish an MSRP session through a relay are more complicated and shown in Fig. 9.1. A relay is needed to allow communication between an insider and an outsider node, i.e., a node in the ad-hoc network and one in the Internet.

The main MSRP specification addresses direct peer-to-peer communication;

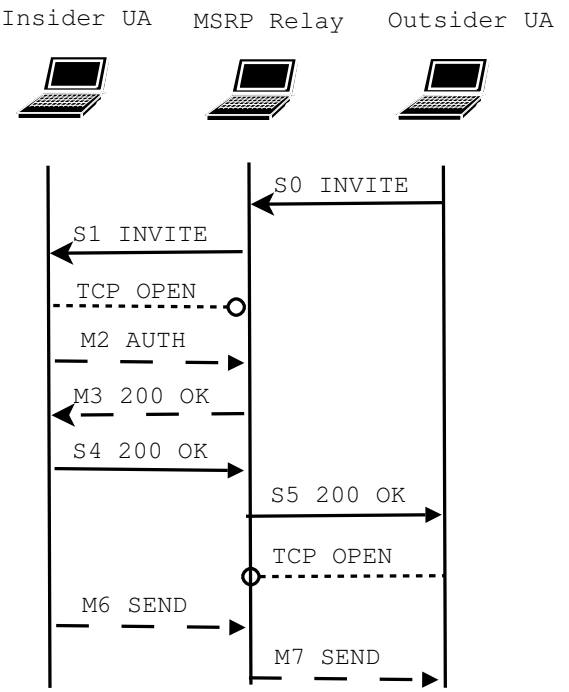


Figure 9.1: MSRP relay usage scenario

however, an extension that uses third party relays when peer-to-peer communication is not possible [15] is being defined. The relay is needed because, in the SIP offer/answer exchange for establishing an MSRP session, the insider node would communicate to the outsider its non-routable IP address as the address where to send MSRP messages. Therefore, any attempt of establishing a TCP connection and directly sending MSRP messages from the outsider to the insider would fail.

The MSRP relay, which can be co-located with the SIP gateway, solves this problem. Although the gateway and relay are co-located, they are two independent entities, listening to two different ports and even using two different transport protocols (UDP at the gateway, TCP at the relay); as a matter of principle, thus, they could run in separate nodes. The address of the MSRP relay can be discovered in several ways; we have chosen to use the SLP protocol adapted for ad-hoc networks [56], similarly to what is done for discovering the gateway address.

The main issue in this scenario is how the insider client realizes its need to use the relay, and how it communicates to the outsider that the MSRP session is not going to be peer-to-peer, but relay based. Let's consider the example of an incoming session invitation, reported in Fig. 9.1. An outsider node sends an INVITE (S0) to an insider; the INVITE is processed by the SIP gateway (colocated

with the MSRP relay). The insider node (namely, the proximity UA extension module) processes the SDP body of the INVITE (S1) and sees from the SDP that the IP address where the remote party wants to exchange MSRP messages is not an ad-hoc address. This implies that the MSRP session must happen through a relay, and the insider must require its services.

According to the MSRP relay specification, a client should be authorized to use the MSRP relay; authorization is carried out by sending to the relay an MSRP AUTH message over a secure transport connection. The mandated way in the specification is TLS, but we decided to establish an IPsec connection between the relay and the insider node. This is possible because the relay and the gateway are co-located. Since SIP signaling and MSRP messaging happen over two different transport protocols, enabling IPSec tunneling allows protecting both signaling and instant messages carrying user data at once. The security solution used to establish IPSec tunnels is a special extension developed within the SESSI project, which will be presented in a journal paper under review at the time of writing this dissertation.

Once a security association with the relay has been established, the MSRP client opens a TCP connection with it (dotted line) and sends an MSRP AUTH request (M2 dashed) to the relay; if the request succeeds, the relay replies with an MSRP 200 OK message (M3 dashed) containing an MSRP URI that the outsider node must use during the session. The MSRP URI contains, among others, the public IP address and TCP port of the relay [15]. The insider adds the received MSRP URI, placing it on top of the MSRP URI list, into the SDP body of the 200 OK response sent to the outsider. The response is routed through the gateway (S4), and finally forwarded to the outsider node (S5).

The SDP body in the S1 invite is similar to the one described in the previous subsection. Now, the UA extensions of the insider modify it to add, in the returned 200 OK reply, the MSRP URI of the relay:

```
   c=IN IP4 192.168.1.143 m=message 1234 TCP/TLS/MSRP
a=accept-types:  message/cpim text/plain
a=path:msrps://relay.example.com:9000/hjdhfha;tcp
msrps://192.168.1.143:1234/fuige;tcp
```

The outsider node checks the topmost MSRP URI and opens a TCP connection towards it. In this case, the list comprises the relay URI and the insider node URI. In a peer-to-peer communication scenario, such a list would have been formed only by the peer URI, as shown in the previous section. Note that the outsider does not need to authenticate with the relay, since it is not requesting its services, but only following the instructions provided by the insider in the SIP signaling exchange. After this step, MSRP messaging happens normally through

the gateway (M6 and M7), using the specified MSRP relay functionalities. In case of outgoing session invitation, the insider node would notice that the target user is not an ad-hoc user, and before sending the INVITE, requests an MSRP URI from the relay, using the procedure described above.

In order to test the described scenarios, we have have designed and implemented on Linux the MSRP relay and the MSRP client, and integrated the clients with the SIP signaling framework. The implementation was tested using laptops connected with each other using WLAN in ad-hoc mode. One of the laptops had a second WLAN card, with a public IP address and acted as the gateway/relay. The ad-hoc clients were modified according to the dSIP architecture, while the registrar server was a standard SIP server. In all the tested scenarios, the SIP and MSRP messaging could get through the gateway/relay, according to the steps described in this section.

A major concern of this approach is whether a mobile node would have enough resources to support the role of SIP gateway and MSRP relay. The gateway node and MSRP relay must in fact run one (albeit simple) additional server and receive, process, and transmit the messages of the ad-hoc users who access the Internet, both signaling and data traffic. This could quickly drain the batteries of a mobile node. The authentication mechanism described in Section 7.5, allows selecting the users to provide the service (e.g, only to the friends whose certificate is known in advance). However, we consider the operator-provided gateway as the most suitable application scenario for a SIP gateway/MSRP relay.

A case of outgoing session invitation would be similar: the UA proximity extensions analyze the INVITE Request-URI, and decide that if this does not match the ad-hoc addressing space then it must be addressed to an Internet node. In such a case, the UA extensions would keep the INVITE message on hold, and begin the MSRP authentication exchange with the relay (we assume that its address is already known. Otherwise it can be searched for before authentication begins) in order to get its MSRP URI. The MSRP relay URI is added before the node's own URI as done for the case of an incoming session. Both insider and outsider node will open a TCP connection to the relay, and send their MSRP messages to it. The relay processes them, properly modifying some MSRP header fields before forwarding them to the intended recipient.

9.4 Advanced Security Services

The security features we have described in Chapter 7 can ensure integrity protection and enforce a message sender's identity in case the two parties involved in communication shared previously a security association with each other, e.g., they had already exchanged the security certificates. During the second part of the

SESSI project we have realized a more general security management framework, allowing also previously unknown users to establish secured session with each other, as briefly mentioned already, and request from trusted third parties one's security certificate.

The SIP service enabling dSIP advanced security features is the Certificate Management framework [49], which is described in Section 2.6.5, adapted for use in a server-less environment. The local server has been enhanced with interfaces to the Authentication and Authorization module in order to fetch and store certificates exchanged using SIP SUBSCRIBE and NOTIFY messages. In the SESSI security framework, users can have a hierarchical structure of certificates, starting from a base certificate and concluding with a series of application-specific (e.g, SIP) certificates. The certificate management scheme allows exchanging using SIP methods any type of certificate, as the certificates are carried transparently in the body of SIP messages, and stored and fetched on the server side using the common API provided by the authentication and authorization module.

We have enhanced the proximity manager application to utilize the security management framework capabilities; the following features are provided to the end-user:

- Exchange of SIP certificates with previously unknown users
- Search of a remote user's certificate
- Request of a third party user's certificate

Exchanging certificates with unknown users can be done using standard SIP messages and the method defined in the Certificate Management Framework [49] adapted for use in ad-hoc networks (i.e., where the entity managing the certificates is the local server). When this option is selected an exchange of SUBSCRIBE/NOTIFY messages is triggered: if e.g., user A requests to exchange the SIP certificate with user B, she will send to B a SUBCRIBE containing the certificate in the body. The reply from B will consist in the 200 OK acknowledging the SUBSCRIBE and in a NOTIFY with B's SIP certificate carried in the body.

Certificates exchanged with this method are stored as non verified in the authentication and authorization (AA) module: the user cannot claim that a remote certificate is valid if in the ad-hoc network there are no third-party trusted entities that can verify it. Non-verified certificates are stored temporarily in the AA module and they are valid only for one communication session; note that securing a session with a non verified certificate cannot ensure the identity of the remote user, because he may be using a forged certificate. The scope of securing sessions with untrusted certificates is to be sure that nobody can tamper with the message exchange of two previously unkown users, e.g., that the identities of the

two communication peers remain confirmed for the whole duration of the session. Of course, an untrusted certificate can be held in the database and verified upon reconnection to the Internet. An even faster solution is to walk and verify physically the identity of the remote user, in a fashion similar to what was suggested by Stajano and Anderson [122].

We utilize the SIP Certificate Management framework to exchange any type of certificate, not only SIP application certificates. The proximity manager application gives the user the possibility of choosing from a predefined list which certificate type should be requested from a remote user. The type of certificate requested is carried in the body of a SUBSCRIBE request, always within the context of the certificate management event package. When the remote server receives a SUBSCRIBE for the certificate management event with a text body, it knows that the transaction does not involve the exchange of SIP certificates as described previously. Instead, it processes the body to find out the type of certificate requested. The format of the body for such an operation is our proprietary design, not standard, that is, this feature can only be used between dSIP enabled devices. The server maps the requested certificate(s) and fetches it (them) from its local database and sends back one NOTIFY message for each requested certificate. The rule that no new NOTIFY can be sent unless a final response for a previously sent request is received still applies here. A NOTIFY contains a single (or none, if there are no locally stored certificates matching the search criteria) certificate and information on the type of certificate being returned with the NOTIFY. The requesting user's server processes the NOTIFY and properly stores in its AA module the certificates, using the API provided. Refining this method is subject of future work and possibly push it for standardization so that it can be used by a wide set of devices, even in infrastructured networks.

If user A trusts user B, i.e., there is a trusted certificate for B in A's database, it is possible for A to request from B a set of certificates from a third user C, if A somehow knows that B holds C's certificates in its database. The search criteria are similar to those described previously, except that the target user for the query is specified to be different from the user to which the request is addressed. With this approach, A can beforehand retrieve and verify C's certificates and begin ad-hoc sessions the following time being sure of C's real identity.

Chapter 10

Experimental Evaluation

This chapter gives an evaluation of the implementation of decentralized SIP. We compare the proactive and reactive operation modes for fully distributed SIP (dSIP), based on the signaling overhead they introduce in the network, for different network sizes. We also compare the overhead of dSIP and sSIP. The overhead that the chosen security mechanism brings about is analyzed against dSIP and sSIP operations. Simulation results for the PCache and SIPCache algorithms are presented as well. Finally, we provide qualitative considerations on the deployment of the SIP gateway and MSRP relay.

The analysis of link-local operations aims at providing a broad idea of the signaling overhead that decentralized SIP introduces in a worst case scenario, i.e., a link-local ad-hoc network where all the nodes are directly connected with each other at link layer and share the wireless medium. In such a network, only one node can transmit at a time; unicast and broadcast communications occupy the shared medium alike. In multi-hop environments, the transmission of a single message is more localized and involves one-hop neighbors only. We consider ad-hoc networks with open users, as this is the worst case in terms of number of signaling messages sent. To have an idea of the global overhead of decentralized SIP, we extend the analysis to big networks with up to 200 nodes. A real life scenario of a link-local ad-hoc network with so many nodes can be a lecture room where a professor explains the lecture topic with the aid of material shared with the students. Note that in such a use case decentralized SIP operations could be modified to give more control and functionalities to the professor's device. There is no need that the students register with each other. Nevertheless, we analyze the overhead of such a scenario as it constitutes the worst case applicability scenario for decentralized SIP.

10.1 Comparison of Proactive and Reactive dSIP

Proactive and reactive dSIP are two complementary ways for enabling the use of SIP in ad-hoc networks. The first method allows the servers to keep their cache updated, and be notified about join, leave and refresh events. In reactive dSIP, cache updates are made only when users issue a session initiation request, but no notifications are sent for presence related events, like arrival or departure of a node. In this section we evaluate the overhead introduced by the two approaches, assuming no security protection. An analysis on the overhead of security protection is carried out later in this chapter.

A comparison between the two approaches is strongly related to the activity rates of the users. In order to quantify the overhead due to dSIP, we need to compute the total number of messages, broadcast and unicast, exchanged in open proactive dSIP to register N users, supposing no message losses in the network. We also hypothesize that no other SIP messages are sent until the N^{th} node arrives.

In open proactive dSIP, when the N^{th} node arrives in the network, it sends a broadcast REGISTER, which will be replied by N-1 200 OK messages, for a total of N messages. Therefore, the overall number of messages exchanged over the air to register N nodes is:

$$M = 1 + 2 + ... + N = \sum_{k=1}^{n} k = \frac{N(N+1)}{2} \tag{10.1}$$

The size of the exchanged SIP messages is variable, since SIP messages include parameters like the AOR, whose length depends on the user name. Based on our tests, made with different user names, we assume an average size of 330 bytes for REGISTER messages, and 310 bytes for 200 OK replies. These values include only the SIP message headers, but not any other underlying protocol. According to Formula 10.1, the number of bytes due to SIP messages sent in the network, after the proactive operations for registering N nodes have been completed, is:

$$B = N * 330 + (\frac{N(N+1)}{2} - N) * 310 \tag{10.2}$$

When N nodes have been registered, in fact, N REGISTER messages are sent in the ad-hoc network, while the remaining are 200 OK replies. Fig. 10.1 is the graphical representation of Formula 10.2. It shows, with the thick line, how many bytes are sent in total as function of the nodes in the network. The trend is parabolic; when, e.g. the 100^{th} node has been registered, around 1.5MB of data have been sent. However, in small ad-hoc networks, with less than 20 nodes, the amount of bytes sent is far below this value, less than 100 KB. Instead, in open reactive dSIP, with N nodes already in the network, every session initiation request triggers N messages over the air. If we suppose a constant session initiation

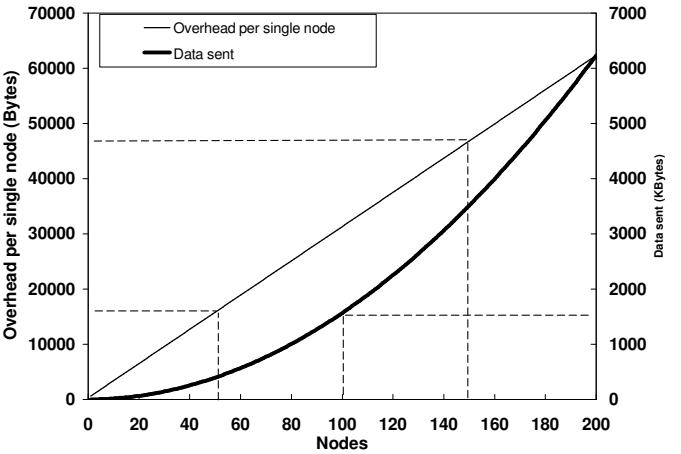
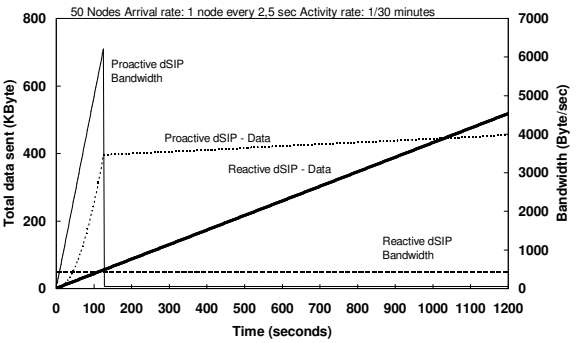


Figure 10.1: Overall and single registration overhead per node of proactive dSIP

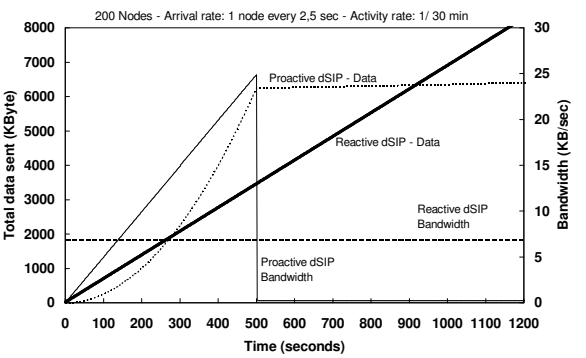
request rate per node, i.e. node activity, the trend would be linear, as we will see later.

Fig. 10.1 shows that, as expected, the price of registering a new node increases with the number of nodes in the network. The thin line traces the amount of bytes exchanged during a single registration operation, with increasing network size. The trend is linear because the difference between two consecutive registrations is always one message (200 OK). The overhead for the registration of a single node, e.g., the $N+1^{th}$, is in fact the difference between the amount of bytes sent in the network to register the $N+1^{th}$ node and the amount of bytes sent to register the N^{th} node. For example, when there are 49 users in the network, the amount of bytes sent in the REGISTER - 200 OK exchange for registering the newcoming 50^{th} user is a bit over 15KB. When there are 149 users, the price of a new registration is little less than 50 KB.

Fig. 10.2 shows a comparison between proactive and reactive dSIP, in terms of bytes sent and bandwidth consumption over a time period of 20 minutes. The bandwidth shown in the graphs was computed considering SIP messages sent in consecutive time intervals of 2.5 seconds, so it can be considered an instantaneous bandwidth. This interval has been chosen equal to the nodes arrival rate in the network, when proactive dSIP was used. For simplicity, we have supposed a constant arrival rate. In a 50 nodes network, there are no longer arrivals after 125 seconds. In a 200 nodes network, arrivals cease after 500 seconds. The total time used for computing bandwidth values, for Fig. 10.2 and the other similar figures in this chapter, does not include external factors like processing time, propagation delay, MAC collisions and especially the back-off interval used by nodes when sending 200 OK replies. This assumption leads us to compute worse values of



(a)



(b)

Figure 10.2: Comparison proactive - reactive dSIP

bandwidth consumption than actually experienced.

The two approaches have been compared for two different sized networks. One is a relatively big network if we consider the scope of dSIP (50 nodes, Fig. 10.2(a)), while the other represents a worst case scenario (200 nodes, Fig. 10.2(b)). The computation for proactive dSIP includes the time and data necessary to register up to N nodes, but after that no join events occur. In reactive dSIP, we suppose that N nodes are already present in the network and have had no dSIP related activities. We compute after how much time, in the defined conditions, the overhead of reactive dSIP overcomes the proactive counterpart.

As to other parameters, the registration refresh interval at each node, which affects proactive dSIP only, was set to 1 refresh sent every 5 minutes. The activity

rate is constant, 1 invitation per node every 30 minutes in Fig. 10.2 (while varying in Fig. 10.3, discussed later). Activity rate of 1 invitation every 30 minutes refers to the number of invitations sent by each node in the network on average. In a bigger network, this implies more messages. Similar considerations are valid for the refresh rate.

The dotted lines in Fig. 10.2 show the amount of data sent in proactive dSIP, while the thick solid lines represent the same value for reactive dSIP. For clarity reasons, the scale of the data and bandwidth Y-axis is different in the left and right parts of the figure. In proactive dSIP the cumulative amount of data sent increases parabolically, as also shown in Fig. 10.1, until all the nodes have been registered, and after that increases linearly since only refresh messages are sent. Reactive dSIP has a lower impact on the network at the beginning, but once proactive dSIP reaches the steady state (all the nodes have been registered), reactive dSIP eventually creates more cumulative traffic. In both network sizes, since the activity rate of nodes is the same, reactive dSIP becomes more aggressive after about 900 seconds, that is, 15 minutes.

Fig. 10.2 also shows the bandwidth consumption of the two approaches in the two networks. The solid thin lines indicate that in proactive dSIP bandwidth consumption increases rapidly in the registration phase, then when the network arrives into a steady state (125 seconds in the 50 node case and 500 seconds in the 200 nodes case), it drops to a very low constant value. For reactive dSIP, bandwidth consumption (dashed lines) is constant, with a value higher than the steady state proactive dSIP.

In any case, the graphs show that dSIP is not excessively aggressive; at its highest peak, for a big network of 200 nodes in fact, proactive dSIP consumes about 25KB/sec. As noticed above, the scenarios analyzed are just examples; however, they provide a reasonable idea on the effective overhead values. Furthermore, we have chosen quite pessimistic values, for all the parameters involved; i.e., we deem that in a real network, an average node arrival rate of 1 every 2.5 seconds is quite high. One example scenario where this rate can be reached is the lecture, if all the students enter the room within a short time frame and register with each other. In the 50 nodes network, which is a much more realistic application scenario for dSIP, the signaling overhead due to registrations has a worst case bandwidth consumption value of only 6KB/sec. Moreover, the scenarios refer to a network of open users; we expect, on the average, that the overhead can be even lower.

The overhead of proactive dSIP maintenance operations is low, as it is possible to see looking at the bandwidth consumption due to refresh REGISTER sending. This effect is depicted in Fig. 10.2 where the bandwidth consumption of proactive dSIP drops to a very low value when the steady state is reached. At the steady

state, no more registration messages are sent by newcomers, and thus no more 200 OK replies, as there are no longer join events in the networks. The only dSIP signaling overhead is due to registration refresh messages.

The overhead computations for reactive dSIP refer to a worst case scenario, where the session initiation requests are all issued by different users. If the same user wants to begin a session shortly after a previous invitation request, the server may assume that the cache entry is valid and avoid triggering a new REGISTER - 200 OK exchange in order to save bandwidth. The price to pay for this is the eventuality of issuing unsuccessful invitation requests. If hybrid dSIP would be used, the refreshes sent after the first REGISTER would prevent the ad-hoc network from being flooded with heavy registrations procedures when a session invitation request is issued.

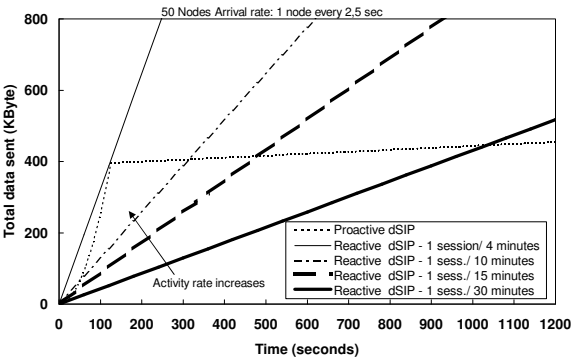
Fig. 10.3 depicts the effect of nodes activity on the overhead of reactive dSIP. The dotted lines and the thick solid lines refer to the same data discussed in Fig. 10.2. When the users are more active, the cumulative signaling data sent in reactive dSIP overcomes earlier the proactive scheme overhead. In a big network of 200 nodes, shown in Fig. 10.3(b), if the invitation interval is about 15 minutes, the overhead of reactive is always higher than the overhead of proactive dSIP. In a smaller network of 50 nodes, 10.3(a), before reactive dSIP becomes permanently more aggressive than proactive dSIP, the activity rate must raise up to 1 invitation every 4 minutes.

In conclusion, from the bandwidth usage point of view, for bigger networks, proactive dSIP is preferable as the overhead of reactive dSIP overcomes the proactive counterpart already for low activity rates. In smaller networks, the reactive approach is preferable. In any case, proactive dSIP has the relevant advantage over reactive dSIP of the immediate bindings availability at the server, which may be a more desirable property for the human user than lower bandwidth usage.

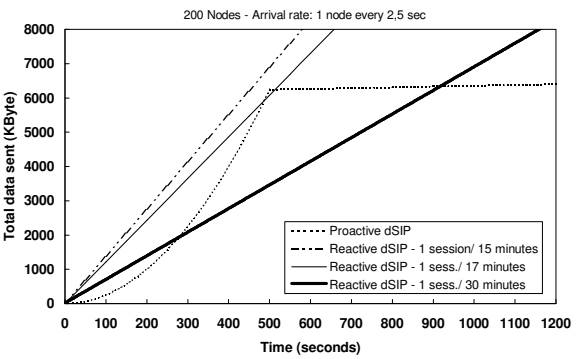
10.2 Arrival and Refresh Rate Effect on the Bandwidth

This section analyzes how the bandwidth consumption varies as a function of arrival and refresh rate, when proactive dSIP is used. We assume a constant nodes arrival rate. Refresh is periodic, and we assume that the refresh period is the same in every node.

Fig. 10.4 shows how the bandwidth consumed varies with different values of node arrival rates, for different sized ad-hoc networks. For clarity reasons, the X-axis is plotted in logarithmic scale; the real increase in bandwidth is linear with the number of nodes. The numbers on the marks refer to the number of arrivals per minute: 60 means that there are 60 arrivals per minute, that is, the arrival rate is 1 node/second. The bandwidth value computed is cumulative, and it is obtained



(a)



(b)

Figure 10.3: Effect of nodes activity on reactive dSIP overhead

by dividing the total number of data sent to register N nodes by the time needed for completing the overall registration, which depends on the arrival rate.

Let us consider the worst case: bandwidth consumption for the highest node arrival rate (1/sec), in the biggest network (200 nodes). We can see that the cumulative bandwidth consumption is around 30 KB/sec, which can be sustained by ad-hoc MAC technologies like 802.11. This value has been obtained dividing the total amount of data sent to register 200 users by the registration time needed when the arrival rate is 1 node/sec. We again recall that, in a real case, the above mentioned concurring factors, such as the back-off 200 OK delays, contribute to make the actual bandwidth consumption value lower than the one computed theoretically.

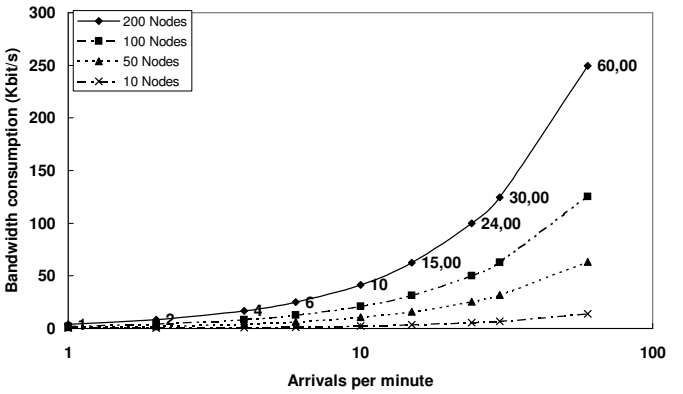


Figure 10.4: Relation bandwidth - arrival rate

The rate of 24 arrivals/minute corresponds to the case discussed in the previous subsection of one arrival every 2.5 seconds. We can see how the values in Fig. 10.1 and 10.4 are related. For 200 nodes, at this arrival rate, from Fig. 10.4, we see that the resulting bandwidth consumption is about 100 Kbit/sec or 12.5 KB/sec. Looking at Fig. 10.1, we can see that for registering 200 nodes about 6MB are needed. If we divide this value for the time needed for registering 200 nodes, at the arrival rate of 24 nodes/minute, that is, 500 sec, we obtain the bandwidth value pointed out in Fig. 10.4.

Fig. 10.5 shows how the bandwidth consumption varies with different values of the refresh rate. The overhead due to refreshes can be neglected, especially compared with the overhead of an original registration. In the worst case of a big 200 nodes network, with fast refresh rate of one message every 5 minutes, the bandwidth consumption value is 220 Bytes/sec. Since this is a constant value, computed in a steady network, this average value of bandwidth correctly coincides with the instantaneous (measured on intervals of 2.5 seconds) value computed, in the same situation, in Fig. 10.2(b). The constant value of bandwidth to which proactive dSIP in a 200 nodes network dropped after registration was 0,22 KB/sec, which is the same that can be inferred from Fig. 10.5. For smaller networks, the refresh overhead is even lower.

10.3 Security Overhead

Protecting dSIP messages for authenticity with dSIP Identity creates an overhead in terms of storage, bandwidth and processing power. This subsection analyzes the overhead that each of these factors bring about.

The dSIP Identity mechanism requires that other users' base and service-

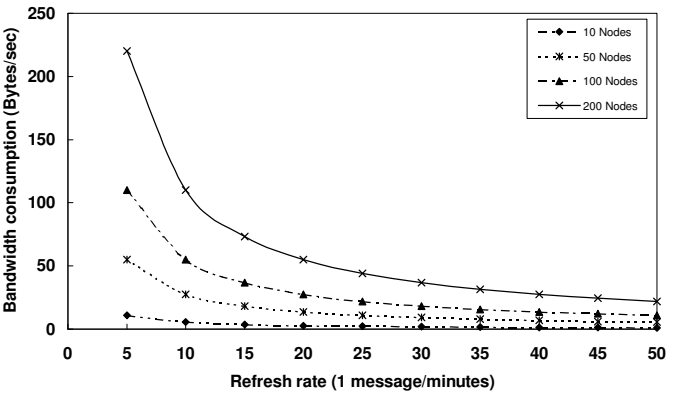


Figure 10.5: Relation bandwidth - refresh rate

specific certificates are stored in the AA module. Storage is not a bottleneck: the size of a huge database with 30000 access control rules and 1000 users, each with 10 service-specific certificates, is about 16MB. Few users will store this much information on this many users, and still, this amount fits easily on almost any smartphone currently on the market.

When using dSIP Identity, the signature and the certificate address, together with the name of the related header fields, add 240 bytes to the message sizes. This value is the same both for requests and responses. Fig. 10.6 compares, in terms of bytes sent, the overhead of registering a new node with a growing number of nodes in the network with and without authentication protection. The thick line corresponds to the values for bytes sent, which were also illustrated in Fig. 10.1; the thin line presents the same values when the security level is authentication (also referred to as integrity). Fig. 10.6 shows that the overhead of a security protected registration operation when the 50^{th} node joins an ad-hoc network is equivalent to an unprotected registration in a network of about 65 nodes. When the network size is closer to the target dSIP network, i.e. 20 nodes at most, the security overhead is practically negligible.

Fig. 10.7 shows the bandwidth consumption for authenticity protected registrations in a 50 nodes network, when the arrival rate is one node every 2.5 seconds. It depicts the same values as shown in Fig. 10.2(a), limited to the time when the last node registers and there is the peak of bandwidth consumption. When the security level is authenticity, the bandwidth consumption is obviously higher than when no security is requested. However, the peak value is only less than 90 Kbit/sec, which is easily sustainable by current ad-hoc technologies. Moreover, the data refer to a worst case scenario and to a network size that is already bigger than the target dSIP network size. These considerations make us safely claim that

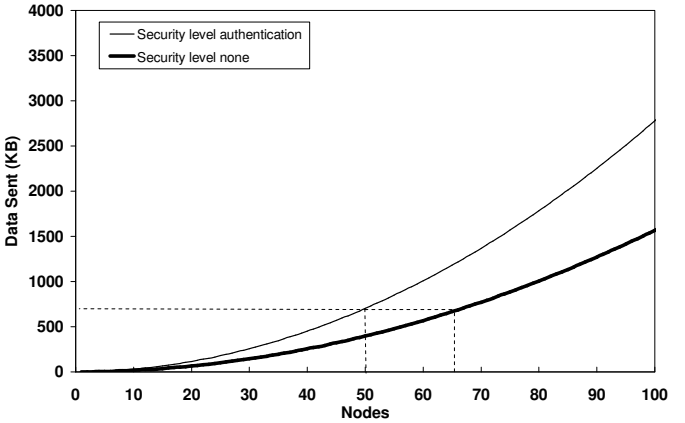


Figure 10.6: Registration overhead for SIP Identity

the bandwidth overhead of the chosen security mechanism is minimal within the design scope of dSIP and that the low overhead design goal was met.

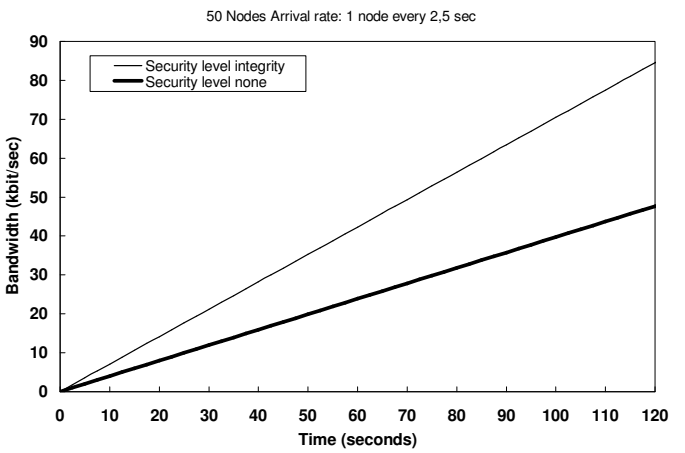


Figure 10.7: Bandwidth consumption of SIP Identity

Security-related processing is caused by locating certificates, mapping certificates to base users, evaluating access control rules and handling signatures. The dSIP Identity scheme uses RSA signatures with 1024 bit keys and even though asymmetric encryption is considered slow, a modern handheld device can create or verify one RSA signature in fractions of a second [3]. Most of the security-related processing is therefore consumed in searching through the database of certificates. We now provide some details on the processing overhead of the AA module; a deeper analysis is out of the scope for this dissertation.

To evaluate the performance, tests were carried out on an IBM T42 laptop computer with 1.8GHz processor running Linux [56]. Table 10.1 details the security-related processing overhead caused by a dSIP Identity protected REGISTER messages and their responses. From there we can see that a refresh REGISTER would cause 0.03s overhead for the sending node, and 0.54s overhead for each receiving node. Incoming messages are remarkably heavier to process than outgoing ones because they require searching through all verified users' certificates.

Table 10.1: AA processing load for Authenticated identity

send REGISTER	30msec
receive REGISTER	540msec
reply to REGISTER	10msec
process reply	540msec

The total security-related processing overhead caused by an arriving node when using the proactive mode in a network of varying size is depicted in Fig. 10.8. We assume a worst-case scenario where all users in the network have authorized each other to receive registration information, and so all nodes reply to an original REGISTER. Sending a REGISTER and processing replies, e.g. from 20 nodes takes about 10 seconds in the test environment. This result is acceptable if we consider that, e.g., Bluetooth authentication can last several seconds and yet Bluetooth has gained wide consumer acceptance.

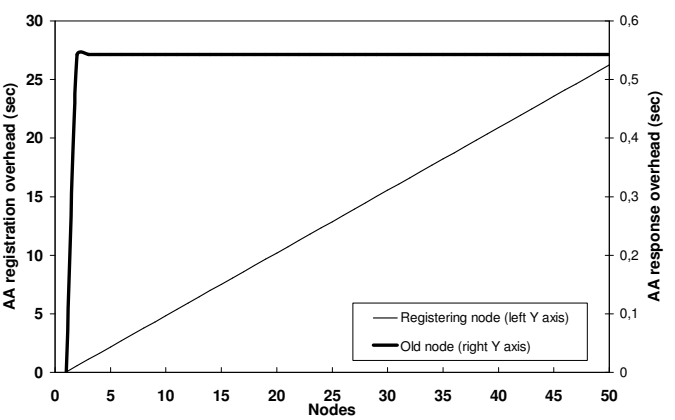


Figure 10.8: Computational consumption for SIP Authenticated identity

For old nodes, the arrival of a new node means that one REGISTER message is processed and replied to, which takes about 0.55s in all. In a handheld device, these times would increase; however, the performance times can be cut remar-

kably with smart database indexing and more refined search algorithms. Additionally, memory cards commonly used on handheld terminals offer much faster read access than hard disks. To further speed up the search, the certificate repository can be read into memory to avoid slow file access from hard disk. To summarize, from Fig. 10.8 we see that, upon entering the network, a node experiences a processing peak due to the registration replies. Nevertheless, once this has passed, the security-related processing load is well within tolerable limits. The overall processing overhead is thereby within the design scope of dSIP.

10.4 Comparison between dSIP and sSIP

Retrieving the bindings of SIP users with SIP-only methods, as discussed in Section 5.8 is complementary to retrieving them using SLP, described in Section 5.9. We carried out a comparison of the overheads due to the two approaches.

When a user registers in an ad-hoc network where there are already N-1 nodes, a broadcast SIP REGISTER message is sent and up to N-1 SIP 200 OK responses are received. With SLP there is logically an exchange of a broadcast query message and several unicast replies returning the bindings. In reality, there is a double phase of broadcast query and unicast replies. In fact, the registering node sends an SLP broadcast Service Query message, to which up to N-1 unicast SLP Service Reply messages are returned. After waiting for a predetermined time, the registering node issues a broadcast SLP Attribute Request query, for retrieving, as service attributes, the AORs associated with the discovered service addresses. While in dSIP the handshake was limited to one sequence of broadcast message and unicast replies, in sSIP up to two times N-1 unicast responses are returned. We consider the worst case, from the bandwidth consumption point of view, that all the other N-1 nodes reply to a registering user when computing the registration overhead of sSIP.

Table 10.2 shows typical SIP and SLP message sizes at transport level, in the various defined security levels. The size of the message depends on the carried payload, like e.g. user or service name; we have measured the sizes with different payload contents, and taken the sizes (bytes) reported in the table as average reference values.

Table 10.2: SIP and SLP typical message sizes

	None		**Integrity**		**Confidentiality**	
	Broadcast	Unicast	Broadcast	Unicast	Broadcast	Unicast
SIP	330	310	570	550	-	-
SLP	110	110	300	290	480	460

In the rest of the discussion, we assume that in SLP there is a single exchange of broadcast messages and unicast replies, where the message size is the sum of the two messages used in each phase, respectively. The value of 300 bytes in the broadcast section for SLP integrity (or authentication) level messages means that the average sum of a broadcast Service Request and Attribute Request is 300 bytes. The same reasoning applies to the other SLP entries of the table. SIP messages are not protected for confidentiality because with S/MIME, which is the most suitable method in our framework for providing confidentiality for SIP, SIP messages would become too big. The bandwidth consumption, and processing load as well, were considered too high compared to the benefits that encryption of SIP signaling messages would produce.

The overhead of a node registration with both approaches increases linearly with the number of nodes in the network. Fig. 10.9 shows the bytes sent over the wireless link to register an incoming node and finding other users in the network, with dSIP and sSIP when no security protection is used, in a worst case scenario where all the other N-1 nodes send a unicast reply to the broadcast message. The increase is linear because when the $N+1^{th}$ node joins the network, 1 broadcast message and N unicast messages are exchanged in the network. So, the increase in bytes is proportional to the average size of a unicast reply, about 310 bytes for dSIP and 110 bytes altogether for the two SLP replies. Expectedly, dSIP is more bandwidth aggressive than sSIP, as SIP messages are bigger than SLP messages. Fig. 10.9 shows that registering e.g. the 50^{th} user with dSIP consumes as much bandwidth as registering the 140^{th} user with sSIP, about 15 kBytes. However, we must bear in mind that a network of 50 nodes is already quite close to (if not bigger than) the scope of our target network environment. For smaller networks, the overhead of dSIP is still quite close to the sSIP counterpart.

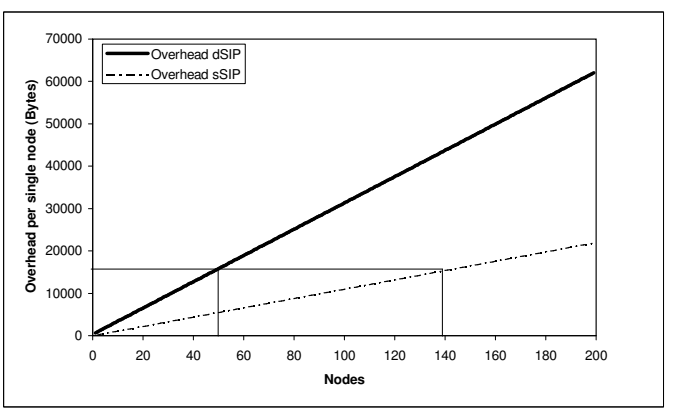


Figure 10.9: Overhead of a single registration. Security level none

Fig. 10.10 shows the bytes sent altogether to register N users. The figure refers to the worst case scenario, with security level of none. We recall that for every registration a number of messages equal to the number of users N in the network are sent; that is, one broadcast and N-1 unicast replies. Consequently, Formula 10.1 holds.

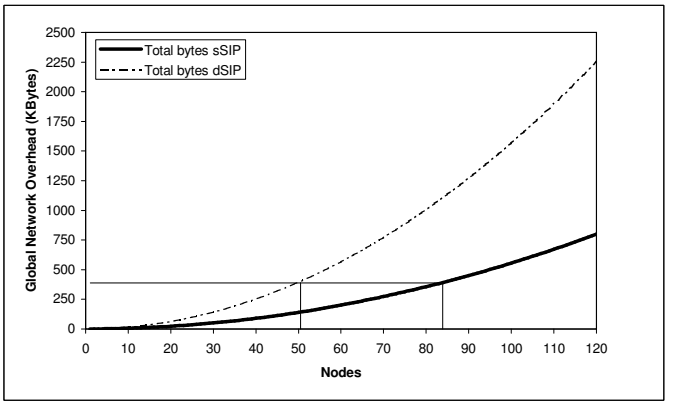


Figure 10.10: Global registration overhead. Security level none

The total registration related bytes sent to register 50 users with dSIP, a little less than 400 kBytes, correspond to the amount of data needed with sSIP to register 85 users.

Fig. 10.11 and 10.12 show the same data when security level is integrity. Both SIP and SLP messages used in this case are bigger. SIP messages have the Identity header fields, which add a overhead of 240 bytes compared to the situation of no authentication; SLP messages carry the additional security block and the signature. Although SIP messages remain bigger, the overhead introduced by the Identity mechanism is lower than its SLP counterpart. The main design goal for secure SIP was in fact to provide integrity with minimal added overhead. Particularly, Fig. 10.11 shows that the overhead of a single registration in dSIP for the 50^{th} node is about 27 kBytes, approximately equal to the overhead of sSIP for a 95 node network. We recall that when no security mechanisms were used, a dSIP network of 50 nodes generated an overhead similar to an sSIP network of 140 nodes. The efficiency, in terms of bytes sent over the air, of the dSIP security mechanism is higher than its sSIP counterpart.

In terms of global registration overhead, it is possible to see from Fig. 10.12 that after the 50^{th} user has registered with dSIP, a total of about 700 kBytes of data has been exchanged in the network. This amount corresponds to the registration of 69 nodes with sSIP. When no security mechanisms are used, the equivalence was reached for an sSIP network of 85 nodes.

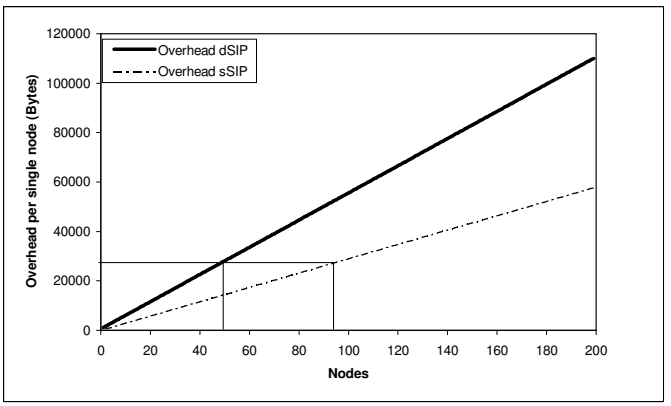


Figure 10.11: Overhead of a single registration. Security level integrity

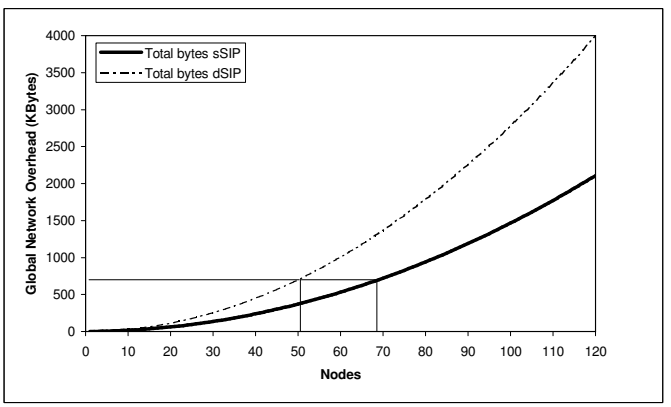


Figure 10.12: Global registration overhead. Security level integrity

10.5 SIP-based Applications for Ad-Hoc Networks

This section provides a qualitative evaluation of the instant messaging and presence services and of the advanced security features implemented to build a proof of concept application that uses the dSIP session management framework.

10.5.1 SIP Gateway Implementation Considerations

Our implementation of the SIP gateway is technically very simple, as it only involves basic almost standard message processing. Indeed, the strength of this approach is its design simplicity; we have extensively researched existing SIP extensions and analyzed new alternatives, in order to find out the solution that best fits the needs of the dSIP framework.

Other solutions could have also been deployed, such as, those addressing SIP NAT traversal techniques [11]. However, they are quite complex to implement, and require support of SIP extensions not fully defined yet. In our approach, the only external entity that needs minor modifications is the registrar server, while the external UA can be a standard one. The insider UA, instead, needs only to be aware of the SIP address of the user to contact: if it does not belong to the ad-hoc network, then the message is routed to the gateway.

An important factor to consider in this design solution is the load that message processing poses on the SIP gateway, also in consideration that the MSRP relay and other services could be co-located in the same node. The type of message processing done by the gateway is not continous, like could be the case of a VoIP media relay; nevertheless, it is important to define policies to enforce access control to the gateway services. In the context of the SESSI project, access control mechanisms have been built as part of the overall service framework, and they are used by the session management service to limit the access to the gateway. For example, people may want to set-up access so that the gateway is utilized only by those users who can present messages signed using a certificate stored as valid in the device AA module.

The SESSI security framework can be utilized also in the case of an operator-provided gateway. In this case, users can either rely on a previous subscription to the operator, and send to the gateway their certificate; the gateway will verify it in the operator's database. Dynamic service can be provided, since dSIP provides means to exchange certificates on the run. In such a case, the truthfulness of the certificate should be verified carefully, to avoid that a user sends forged authentication data. In the case of an operator provided gateway, attached with the other interface to the fixed network and not battery powered, there is less reason to worry about load on the gateway as for the case of gateway being a mobile node.

10.5.2 Overhead of the IM and Presence Framework

This section analyzes the network overhead, in terms of bytes sent, of the Instant Messaging and Presence framework messaging, with and without security support. We assume that the user has already spread a registration in the network, and got the list of available users. He decides to subscribe to the presence state of one of them and then to begin an MSRP instant messaging session.

Table 10.3 shows the total header field sizes in average for messages used in the presence framework and the instant messaging using MSRP, with and without security support. The values are only approximations because the messages include variable information, such as, the URI of the parties, and the type and name of the presence information. When a user subscribes to the presence state of another user, an average of 630 bytes will be exchanged due to the SUBSCRIBE

Table 10.3: Message sizes list in the IMP framework

	Unsecured	Secured
Presence Subscription	630	1070
Presence Notification	1340	1820
Presence Status Change	1340	1820
SIP Session Establishment	1600	2320
MSRP Messaging	390	–
SIP Termination	440	920

- 200 OK response. If the session is protected, then additional 480 bytes (240 for each exchanged message) are needed.

A presence notification is much heavier, since the NOTIFY message contains the presence document of the notifying entity. The data format chosen by IETF for representing the presence data is called Presence Information Data Format (PIDF) [124], which is XML-based, and therefore creates quite big messages, which can easily exceed 1KB. A notification in the presence status of an entity causes the whole presence document to be delivered again to the subscribed entity; the entire XML document is delivered even though only part of the presence information has changed. This behavior is quite inefficient, especially in wireless environments or when the devices have limited resources. It is no surprise, therefore, that the IETF is developing a solution for communicating only the pieces of presence information that have changed, in order to limit the bandwidth consumption of presence notifications [76]. We assume, again to consider the worst case, that no partial notification of presence information is used; therefore, a presence update exchange is as expensive as the original notification.

The SIP session establishment bytes take into account the three-way handshake INVITE - 200 OK - ACK, but not the provisional responses exchanged during the session establishment process. MSRP messaging is a highly variable part in the process, as the actual bytes depend on the content of the instant messages exchanged by the users. The entry in the table comprises also the (MSRP) 200 OK reply returned by the receiver of a MSRP message; sending this response is optional, but we add it here for completeness. Note that an MSRP session cannot be protected for security with the SIP Identity method, which only applies to SIP signaling. In any case, MSRP is the media signaled by SIP, totally independent from it, so applications using MSRP can choose any method to secure the session (e.g., MSRP over TLS). The SIP termination phase comprises the BYE request and the 200 OK response.

The total number of bytes exchanged in this scenario is therefore slightly over 5KB for unsecured sessions, and almost 8KB for secured ones. The numbers do

not comprise MSRP messaging. Each MSRP instant message has a fixed overhead due to the headers, and eventually the 200 OK response; an average value for this overhead could be 390 bytes. The amount of bytes sent over the network for a messaging session grows linearly for each message. For example, if two people send 20 chat messages of 100 characters each, the whole session results in about 10KB sent over the ad-hoc network. A long discussion with 100 messages of 500 bytes would consume about 100 KB.

If we consider that the messages are exchanged over an extended amount of time, and that ad-hoc networks formed by WLAN technology can reach bit rates of several Mbit/sec, we can see that the overhead of SIP-based IMP is quite reasonable. We underline the importance of having a standard for IMP, in any network environment, so that users could use the same service no matter if they are in ad-hoc networks or infrastructured environments. Deploying an IMP solution especially tailored for ad-hoc networks may probably be better from the bandwidth usage point of view, since messaging could be optimized for such an environment, but would drastically break interoperability with other IMP systems, de facto seriously hindering the usability of the system itself. Using a standard framework like the one provided by SIP solves this problem.

10.6 Evaluation of dSIP for Multi-Hop Environments

This section provides and discusses results for simulations run to evaluate the performance of the algorithms devised to deploy dSIP in multi-hop ad-hoc networks.

10.6.1 Simulation Results

A prototype of PCache was implemented in the *ns-2* network simulator v 2.28. The simulated network is composed of 100 nodes uniformly distributed over a region of 1500mx500m. Nodes move accordingly to the random way-point model [52] using three different speed models: 0m/s, 3-7m/s and 5-15m/s. In the latter cases, pause times are randomly selected between 0 and 20s. 10 movement files were randomly generated for each speed. The simulated network is an IEEE 802.11 at 2Mb/s. Network interfaces have a range of 250m using the Free Space propagation model. A real life situation corresponding to the simulated scenario could be a set of rescuers spread over an area where a natural cataclysm has occurred.

Runs are executed for 900s of simulated time. Each run consists of 100 disseminations and 400 queries. Each node disseminates one data item in a time instant selected uniformly from the time interval between 0 and 400s. Items are uniquely identified by numbers between 0 and 99. Simulations do not consider expiration of cache entries since they could unfairly improve the performance of

the protocol by freeing additional resources on the caches of the nodes. Queries start at 200s and are uniformly distributed until 890s of simulated time. The nodes performing the queries and the queried items are randomly selected. The simulation ensures that only advertised records can be queried so that the evaluation of the protocol does not become obfuscated by bogus queries. No warm up period is defined and 10 traffic files were generated.

The sensitivity of PCache to different parameters is evaluated by testing the parameter with different values while keeping the remaining parameters consistent with the baseline configuration. In the baseline configuration, the cache of the nodes was defined for accepting at most 10 items, excluding owned items, which are stored in a separate region of the memory. Each data item has a size of 250 bytes (50 for the key and 200 for the value). In this configuration, a full cache occupies about 3KBytes, which is a small value. Bigger values of cache size improve the performance of PCache, as there is more storage space available in the network. The message size was limited to 1300 bytes. After removing the space required for the PCache header (estimated to be 13 bytes for the fixed part), a PCache message will carry at most 5 data items.

Unless stated otherwise, all values presented below average 10 independent runs, combining one movement file and one traffic file. Although in the rest of the evaluation we refer to PCache experiments, we underline that the evaluation is valid for SIPCache as well, since SIPCache is an application use case of the general PCache algortihm. The performance of the algorithm is evaluated using the following metrics:

Average distance of the replies (DR) Averages the distance, in number of hops, from the querying node to the source of the first reply received. The average distance of a reply will be 0 if the value is found in the cache of the querying node.

Average nodes without an item in 1-hop neighborhood (N1) At the end of the simulation, and for each data item d, accounts the number of nodes that do not have a replica of d in its 1-hop neighborhood. The value is the average of this count for the 100 data items. This metric was only evaluated for scenarios with speed 0.

Number of dissemination/query/reply messages and bytes This metric measures the number of messages/bytes sent to the network for each PCache type of operation. Each forwarding by a node is accounted as one message and contributes with the size of the message (at the MAC level) to the total number of bytes.

10.6.2 Sensitivity to Probabilities

Tables 10.4 to 10.7 present the DR and N1 metrics for different combinations of p_{ins} and p_{rep}. These constants, which have been introduced in Section 5.11.7, dictate respectively the probability of having a Complementary item to be inserted in a node's cache and the probability of replacing a Complementary item before forwarding a message.

p_{ins}/p_{rep}	0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.97	0.97	0.98	0.98	0.98	0.98
0.2	0.97	0.96	0.98	0.98	0.98	0.99
0.4	0.97	0.98	0.98	0.98	0.99	0.98
0.6	0.98	0.99	0.98	0.98	0.99	0.99
0.8	0.97	0.98	0.97	0.98	1.00	0.99
1.0	0.97	0.97	0.98	0.99	0.99	1.00

Table 10.4: Average distance of the replies for speed 0ms^{-1}

p_{ins}/p_{rep}	0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.97	0.96	0.96	0.97	0.98	0.99
0.2	0.95	0.96	0.96	0.96	0.97	0.97
0.4	0.97	0.96	0.96	0.97	0.96	0.98
0.6	0.96	0.98	0.96	0.96	0.98	0.97
0.8	0.97	0.95	0.96	0.98	0.96	0.97
1.0	0.97	0.97	0.96	0.98	0.98	0.99

Table 10.5: Average distance of the replies for speeds 3-7ms^{-1}

p_{ins}/p_{rep}	0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.98	0.97	0.96	0.98	0.98	0.98
0.2	0.96	0.95	0.96	0.96	0.97	0.97
0.4	0.98	0.95	0.95	0.98	0.96	0.97
0.6	0.97	0.97	0.97	0.97	0.97	0.98
0.8	0.96	0.97	0.96	0.97	0.97	0.98
1.0	0.97	0.98	0.96	0.96	0.98	0.98

Table 10.6: Average distance of the replies for speeds 5-15ms^{-1}

All values for the DR metric are in the interval $[0.95, 1.0]$. Because the distance between the extremes of the interval is 5%, it can also be said that PCache is not particularly sensitive to the values of these probabilities. The tables show

p_{ins}/p_{rep}	0.0	0.2	0.4	0.6	0.8	1.0
0.0	8.50	8.69	8.78	8.74	8.22	8.68
0.2	8.97	9.26	9.35	9.18	9.21	9.21
0.4	9.42	8.81	9.37	9.48	9.31	9.78
0.6	9.54	9.53	9.06	9.39	9.28	9.96
0.8	9.95	9.50	9.64	9.79	9.60	9.92
1.0	9.95	9.33	9.83	10.09	9.53	10.17

Table 10.7: Metric N1 for different combinations of probabilities

that for some combinations of p_{ins} and p_{rep}, PCache performs better when nodes move. This effect is attributed to the tendency of the random way-point model to concentrate nodes at the center of the simulation space [10]. Since node density increases at the center, so does the probabilities of having at least one of the neighbors with the values requested in cache. For this reason, the evaluation of experimental results will privilege the case where nodes do not move.

The best values for the DR metric are not coincident with those for the N1 metric. It should be noted that the results provided by DR depend on the random pattern of queries and directly affect the distribution of the items, specially because a querying node will store the reply in its local cache. On the other hand, N1 is a *post-mortem* analysis of the distribution of the items, which has already been biased by the queries performed to measure N1.

The results suggest that low probabilities are preferable. Contrary to the intuition, the use of high probabilities does not increase the heterogeneity of the caches. Instead, it prevents nodes from disseminating their records beyond their 1-hop neighbors. The *storedInSender* flag prevents the receivers of a message from storing the records inserted in the previous hop but a high value for p_{rep} mandates the nodes to replace them before retransmission, restarting the loop. The possibility of not including Complementary records in the messages has different undesirable effects, that should be prevented. It slows the speed at which joining nodes will fill their cache, therefore worsening the performance of the protocol. Furthermore, even when not inserted or replaced in messages (as is the case when $p_{ins} = p_{rep} = 0.0$), Complementary records are used by the 1-hop neighbors of the sender to learn which records are redundant, and making them candidates for replacement.

Values of $p_{ins} = 0.4$ and $p_{rep} = 0.2$ present the most consistent combination of results from both metrics and for all speeds. These values were selected for further analysis and are those used in all tests presented further.

10.6.3 Analytical Evaluation of the Number of Copies Stored

To make the analysis of the number of copies stored by the dissemination algorithm manageable, it was necessary to impose some constraints to the modelled environment. It is assumed that each execution of the dissemination algorithm progresses in rounds. That is, each node considering to transmit in round r has received the original transmission and any retransmissions in round $r-1$ and that all nodes transmitting at round r can not influence the decision to transmit of any other node that has also received the transmission in round $r-1$. The analysis will consider only the nodes that decide to retransmit since the remaining do not actively contribute for the outcome and do not influence the decision of nodes that retransmit.

The number of copies of each data item stored in the network during a dissemination phase in PCache is estimated using three interdependent functions.

$T(r,t)$ for some message received by a node in round r, returns the probability of the message having Time From Storage TFS t.

$m_T(r,t)$ gives the probability of mintfs[1] being t for a node in round r that at the end of the holding period has decided to retransmit;

$S(r)$ returns the probability of a node in round r to have stored the item;

Consider a node in round r that received a message m transmitted by node s in round $r-1$. The TFS of the message will be 0 if s stored the item and 1 or 2 otherwise.

$$T(r,t) = \begin{cases} S(r-1) + (1-S(r-1))m_T(r-1,2), & t = 0 \\ (1-S(r-1))m_T(r-1,t-1), & 1 \leq t \leq 2 \end{cases} \tag{10.3}$$

Node s may have stored the item for two reasons, each represented by a parcel of the sum in the first row of Eq. 10.3. The first parcel of the sum gives the probability that the item had been stored as a result of the randomness of the algorithm and the second the case where the item was stored because mintfs was 2. The second row considers the probabilities of the previous node not having stored the item. Algorithm rules dictate that the value advertised in the TFS field should be one above the mintfs determined by s. Therefore, the probability of receiving m with TFS t is the same as the probability that s did not stored m and had a mintfs of $t-1$.

[1] We recall that mintfs is the minimum value of TFS listened by a node during the period when a received dissemination message is put on hold. See Section 5.11.4.

The mintfs value of each node depends on the TFS of the transmissions of m it receives. All the possible situations are arranged in four cases, relevant for the exposition:

1. The node received only the original transmission. In this case, mintfs will be the value of this message;

2. The node received the original transmission and one retransmission. In this case, mintfs will be the lowest TFS of both messages;

3. The node received any number of messages, all having the TFS field set to two. For the purposes of this analysis, this can be considered a particular case of the previous one, when the first two messages received had TFS$=2$;

4. The node received at least three messages, the original and two retransmissions and at least one of them had TFS lower than 2. In these conditions, the node will not store or retransmit the data item, therefore, this case is not relevant for the analysis.

Function m_T is defined considering the first two cases. Since the number of messages depends on the network topology, and can change, for the same topology for each source node, we assume an equal probability of the holding period to expire after the node had received one or two messages. m_T is defined as:

$$m_T(r,t) = \begin{cases} 1, & r = 1 \wedge t = 0 \\ 0, & r = 1 \wedge t \geq 1 \\ \frac{T(r,0)}{2} + \frac{T(r,0)(T(r,0)+2T(r,1)+2T(r,2))}{2}, & r > 1 \wedge t = 0 \\ \frac{T(r,1)}{2} + \frac{T(r,1)(T(r,1)+2T(r,2))}{2}, & r > 1 \wedge t = 1 \\ \frac{T(r,2)}{2} + \frac{T(r,2)T(r,2)}{2}, & r > 1 \wedge t = 2 \end{cases} \tag{10.4}$$

mintfs is only evaluated for rounds after the initial dissemination. Therefore, the function is not defined for $r = 0$. The results of the function for the first round are immediate: in round 0 only one message is transmitted, with TFS$=0$. Therefore, mintfs for nodes in the first round must be zero. Other values of mintfs will occur with probability zero.

For rounds other than the first, m_T evaluates the probability of receiving all the permutations of 1 or 2 messages and arranges them accordingly to the mintfs of each permutation. We remind that the probability of receiving each individual message is given by $T(r,t)$. Since it is assumed that retransmissions in the same round are independent, the probability of receiving some pair of messages is given by the product of the probability of receiving each of them. In Eq. 10.4, for each t,

the first component of the sum accounts the probability of receiving only one message, with TFS$=t$. The second component considers all possible combinations of pairs of messages, for which the minimal TFS will be t. Each of the components is divided by two to distribute equally the probabilities between both cases.

With the auxiliary functions defined, it is now possible to proceed to the definition of $S(r)$ which will return the probability that a node that forwarded a dissemination message in round r, has also stored the item in its local cache. $S(r)$ is presented in Eq. 10.5

$$S(r) = \begin{cases} 1, & n = 0 \\ \sum_{t=0}^{2} e^{t-2}.m_T(r,t), & n > 0 \end{cases} \tag{10.5}$$

$S(0)$ is 1 to comply with the requirement that the source node always stores the items it advertises. For nodes in other rounds, the probability will be given by weighting the probability of the occurrence of each mintfs with the exponential function introduced in the algorithm description.

Fig. 10.13 applies $S(r)$ to the first 20 rounds. As expected, it can be seen that the algorithm creates a local maximum at every third round followed by a local minimum. As the message progresses for more distant rounds, so the differences of the probabilities become more attenuated, to attend to the wider region covered at each round.

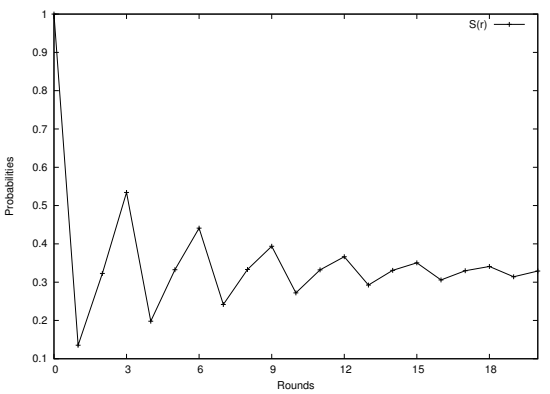


Figure 10.13: Probability of storage for nodes located at different rounds

This section has shown that the algorithm adapts the storage probability to the distance to the source of the item. The adequacy of this algorithm for retrieving items will be experimentally evaluated in the following section. To conclude the analysis we notice that the number of replicas expected to be stored at each round could be derived by combining $S(n)$ with the number of retransmissions on that

round. This is deferred for Section 10.6.4 where the number of messages collected from a simulation will be used to estimate the number of cached copies of data items.

10.6.4 Evolution of the Number of Copies

If the location of the nodes was ignored, the desirable distribution of items would divide equally the total cache space (given by the sum of the cache space of all nodes) by the number of items. For the baseline configuration presented above, and considering also the copy stored at the source node, each item should have 11 copies. Figg. 10.14 and 10.15 show the evolution of the number of copies of some items and of the standard deviation of the number of copies of all items in one representative run of the tests above, with speed 0m/s, $p_{ins} = 0.4$ and $p_{rep} = 0.2$. In this particular run, the metric DR evaluated to 0.98 and the N1 metric to 7.85. The pattern exhibited in these plots is similar to that observed in the remaining tests.

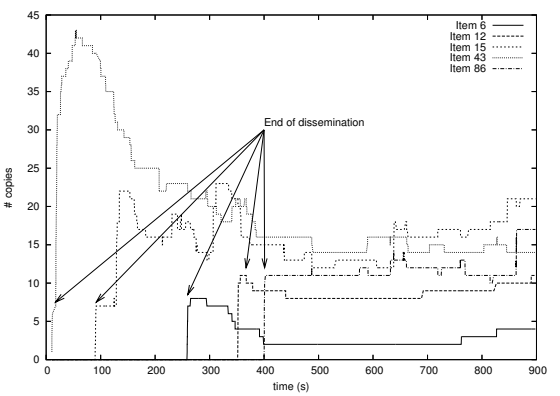


Figure 10.14: Evolution of the number of copies in one simulated run of PCache

The data items represented in Fig. 10.14 are those with the lowest (item 6) and highest (item 15) number of copies at the end of the simulation, of the first (item 43) and last (item 86) items disseminated and of one of the items that reached the end of the simulation with the average number of 11 replicas (item 12). All of them exhibit a common pattern which can also be found in a large majority of the items in all runs. In all of these cases, the number of replicas rises quickly from 0 to some value usually close to 8. From the inspection of the trace files, it was concluded that this fast climb is due to the dissemination phase. Later variations are a result of changes in the caches triggered either by Complementary items or the storage of records in querying nodes.

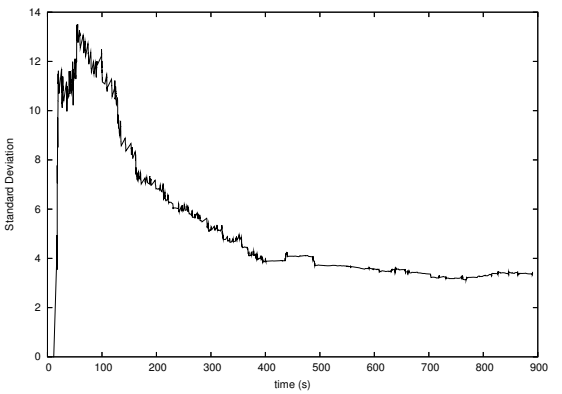


Figure 10.15: Evolution of the standard deviation in one simulated run of PCache

Table 10.8 applies the analytical model defined for PCache in Section 10.6.3 to the run. The number of forwards on each round presented in the third column was extracted from the trace file of the simulation. The last column applies the function $S(r)$ to the values observed for the number of retransmissions. As it can be seen, the total number of replicas expected to be stored is compatible with the experimental results, for example with those presented in Fig. 10.14.

Round	S(r)	# Transm.	# Copies
0	1.000000	1.00	1.00
1	0.135335	5.52	0.75
2	0.322802	5.51	1.78
3	0.534140	4.33	2.31
4	0.198021	3.55	0.70
5	0.332449	2.77	0.92
6	0.441114	2.01	0.89
7	0.241633	0.90	0.22
8	0.333138	0.31	0.1
9	0.393713	0.05	0.02
Total	-	25.94	8.68

Table 10.8: Average number of messages per round

The incorporation of the item in the Complementary items section of other dissemination messages justifies the large number of copies attained by the first item advertised in the simulation (item 43, in this run) and consequently, of the rapid

increase in the standard deviation. Because the cache at the nodes is not filled, nodes forwarding the dissemination message of other items store the complementary items independently of p_{ins} (although the constraint of the storedInSender flag still holds). However, it can be seen that as the number of advertised items grows, the number of copies of the first item decreases, suggesting that PCache is able to balance the number of copies of each data item. The plot of the standard deviation of the number of replicas of the items, shown in Fig. 10.15, supports this claim. After an initial growth it tends to stabilize around a small value (approximately 3.5) as soon as registrations finish.

Fig. 10.16 is an histogram of the number of copies of each data item at the end of the simulation. For each data item, the graph presents the number of copies in excess (positive) or missing (negative) to the average number of copies. The distribution seems to be balanced, with more than 50% of the nodes in the interval $[-2, +2]$ and not presenting a significant number of peaks either for the positive or negative side.

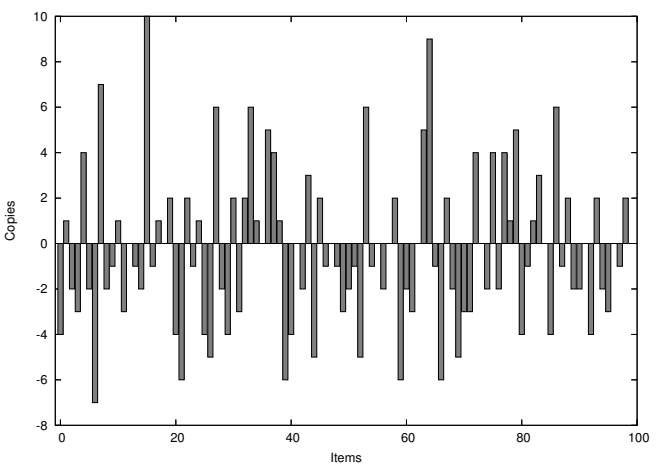


Figure 10.16: Difference to the average number of copies

Fig. 10.17 shows in black the geographical distribution of the item with the minimum number of copies at the end of its dissemination phase. It can be seen that the algorithm was not able to put a copy near every node in the network. However, it was able to spread copies of the item over geographically distinct points of the simulated region.

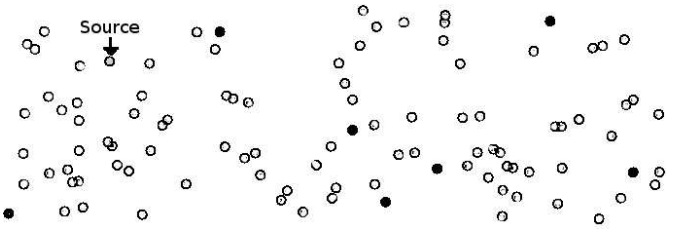


Figure 10.17: Dissemination of the item with the lowest number of copies at the end of the simulation

10.6.5 Impact of Item Size

The impact of the size of the data item on the average distance of the replies is presented in Fig. 10.18. In this set of experiments, the size of both the key and value components of the data item was increased while keeping the maximum size of the message constant. This effectively results in varying the number of items present in a message. This test case also covers the scenario of SIPCache messages protected for integrity. An integrity-protected SIPCache message is bigger in size than an unprotected one, due to the addition of the signature. Since when SIP messages are exchanged over UDP, as is the case of SIPCache, the maximum size of the SIP message is limited by the specification to 1300 bytes, the result is that less data items can fit into an integrity-protected SIPCache message compared to the unprotected case.

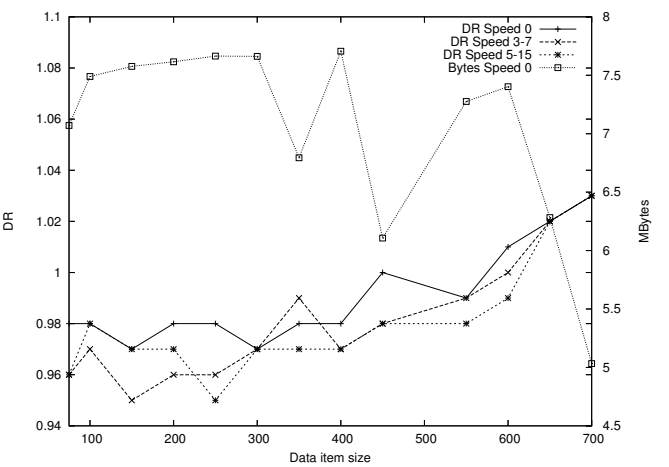


Figure 10.18: Variation of metric DR with the size of the data items

Results for the DR (average distance of replies) metric show that the performance of PCache degrades as the size of the data items increase, i.e., less data items fit into the body of a message. The total number of bytes transmitted by all nodes in the simulations for speed 0 is represented in the right y axis to justify that the performance degradation can not be attributed to the increased bandwidth consumption. (The total number of bytes transmitted in other speeds follows a similar pattern but is slightly lower, and it is not shown for clarity.) Instead, we justify this behavior with the gradual decrease of the number of complementary items that can be inserted in messages. For values higher than 600 bytes and messages with maximum size of 1300 bytes, complementary records can only be carried in query messages, because only the key of the item to be retrieved is sent.

An excessive number of items in a message may also prevent complementary items from performing their role on leveraging the item distribution. As shown by metric N1 in Fig. 10.19. For a large number of data items (i.e. when the data item occupies a small number of bytes), the absolute number of records that are inserted in cache and replaced in forwarded messages (dictated by probabilities p_{ins} and p_{rep}) increase and begins to adversely affect the cache distribution.

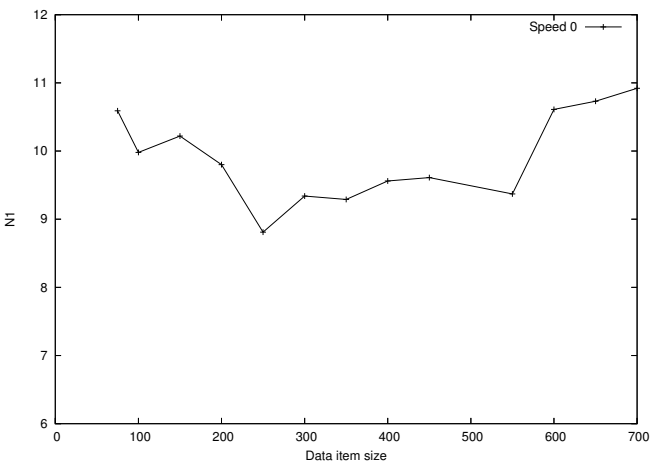


Figure 10.19: Variation of metric N1 with the size of the data items

10.6.6 Sensitivity to Cache Size Ratio

To reduce query traffic, PCache aims at storing a significant proportion of the data items in the 1-hop neighborhood of every node. The number of cache entries available in the nodes located in the 1-hop neighborhood is conditioned by the size of the cache of each node and the number of neighbors.

Relevant for this analysis is the ratio between the size of the cache in the 1-hop neighborhood and the total number of items. This ratio, hereafter named Relative Neighborhood Cache Size (*RNCS*), is given by $RNCS = \frac{\Sigma_{i \in \text{neigh}} CS_i}{\#items}$ where CS_i is the cache size of node i in a neighborhood and *# items* is the total number of advertised items in cache. Assuming that the number of 1-hop neighbors is n and that all nodes have an equal cache size, the equation can be simplified to $RNCS = n\frac{CS}{\#items}$. The effect of the variation of each of these parameters in PCache was evaluated individually. In each of the three tests, the remaining parameters were kept according to the baseline configuration described in the beginning of the section. All results presented in this section are the average of 10 runs for each condition described.

Figg. 10.20 and 10.21 label the variations of *RNCS* by changing the cache size of each node and of the total number of items respectively as "Cache Size" and "Items". For the variation of the cache size, the results were computed using sizes between 3 and 17 items at intervals of two. The number of items varied in the interval between 50 and 400 at intervals of 50.

To vary the number of neighbors, the baseline configuration was tested with the nodes configured for transmitting with different ranges while keeping the size of the simulated space constant. The tests were performed for transmission ranges between 150 and 325 meters at intervals of 25 meters.[2] The number of neighbors was estimated by averaging the number of nodes that received every broadcast message on every simulation with the same transmission range. The figure identifies the results of these tests with the label "Neighbors".

The resulting DR metrics are presented in Fig. 10.20. As expected, performance of PCache increases with RNCS. For values of RNCS close to one, replies to queries will be found on average at 1.3 hops and at values close to 1.8 hops as the RNCS approaches 0.5. It should be noted that the values for RNCS equal to 0.5 where obtained by changing the number of data items to 400. Because the remaining parameters are always kept according to the baseline configuration, each individual node was able to store 3.5% of the total number of items.

The figure reveals that the RNCS values are not coincident for the different speeds, even when tested with the same parameters. Again, we attribute this to the tendency of the random way-point movement model to concentrate nodes at the center of the simulated space [10], thus increasing the number of neighbors. The RNCS value corresponding to the baseline configuration at both speeds are highlighted in the figure. We emphasize also that a comparison between similar tests with different speeds is not immediate because for the same resources, different speeds result in different values of RNCS.

[2] Transmission ranges below 150m do not provide accurate results due to the large number of isolated nodes.

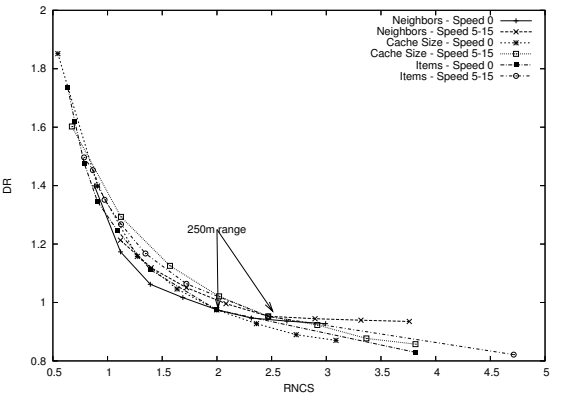


Figure 10.20: DR metric for variations of RNCS

Because one of the criteria used to change RNCS makes the number of neighbors vary, the results for the N1 metric can not be compared. Therefore, Fig. 10.21 presents N1 after a normalization which consisted in dividing the results by the average number of neighbors. The results are consistent with those presented for DR. The figure shows that PCache can handle better a reduced number of neighbors but the benefit of adding new nodes has a limit when the total capacity of the neighborhood reaches twice the number of advertised items. On the contrary, with the increase of the size of the cache or the reduction of the number of items, PCache is capable of further improving its performance.

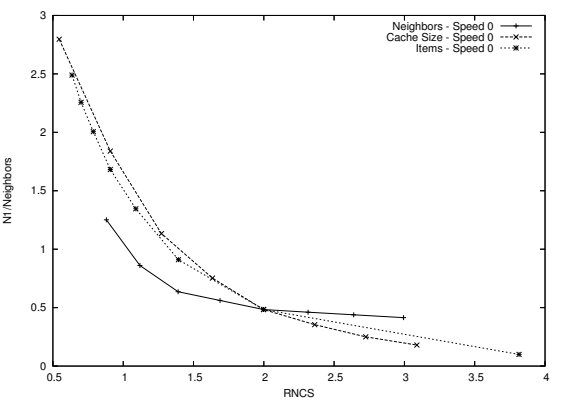


Figure 10.21: N1 metric for variations of RNCS

10.6.7 Number of Messages

Fig. 10.22 shows the average number of transmissions per PCache operation in the tests of section 10.6.6 that changed the transmission range. The number of nodes in the network was kept unchanged. The number of messages per event account, for each type of message, the original transmission and all retransmissions performed during the execution of the broadcast algorithm or every forwarding of a point-to-point message. Although sharing a common broadcast algorithm, the figure shows a significantly smaller average number of messages of type query. This is due to the efficiency of PCache in item distribution.

Part of the queries are replied locally and do not require a broadcast, while others may be replied by some of the 1-hop neighbors and therefore, require a single transmission. The decrease in the number of messages as the number of neighbors increases results from distinct factors for dissemination and query operations. In disseminations, it is due to the restriction imposed on the number of retransmissions.

As the number of neighbors increases, so does the number of nodes capable to receive each retransmission and, therefore, do not retransmit. This factor also affects the flooding of query messages. However, as fig. 10.20 has shown, the probability of finding the queried item within the 1-hop neighborhood increases for higher density of nodes, thus reducing the number of floodings of queries performed. Finally, the figure shows that after an initial decrease, the number of reply messages tend to stabilize, due to the excessive redundancy of replicas that leads to an increasing number of replies for each query.

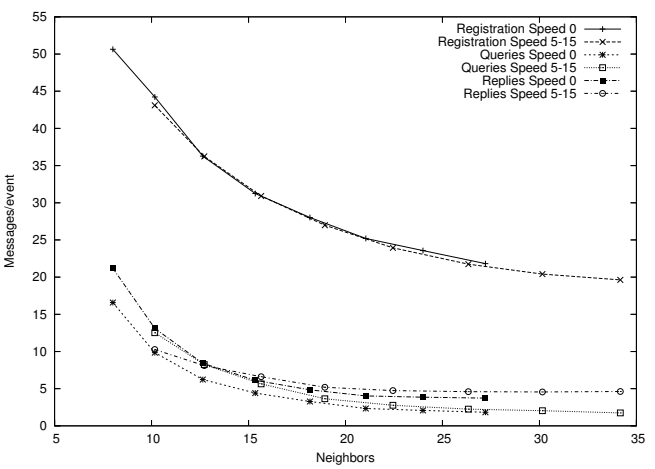


Figure 10.22: Number of messages per operation

The effect of two expanding ring search policies on the number of queries and reply messages is compared in Fig. 10.23. For clarity, the results for the intermediate speed are omitted. The figure shows that performing the first step of an expanding ring search with TTL=2 is preferable when RNCS is low. These results were collected by changing the transmission range, in the tests described above. In this scenario, there is a low probability of having a query replied in the 1-hop neighborhood and, to prevent the subsequent flooding, it is more advantageous to perform an initial extended search, which has an higher probability of having the query replied. As RNCS increases, the number of queries replied with TTL=1 gain dominance. The propagation of the query to the second hop is useless in the majority of the cases and consumes additional resources both on the unnecessary propagation of the query and of redundant replies.

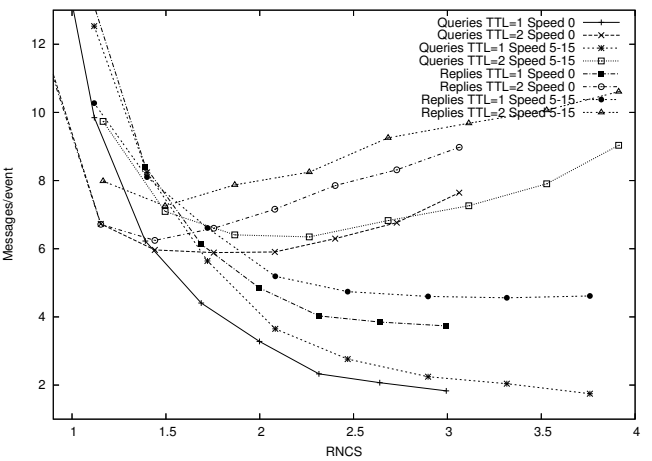


Figure 10.23: Comparison of the number of messages for queries and replies for TTL=1 and TTL=2

10.6.8 Profile Based Queries

In this section we refer explicitly to SIPCache, as Profile Based Queries have been introduced to meet the requirements of PCache when used as a SIP Location service in ad-hoc networks. The performance of SIPPCache in profile based queries is evaluated by the proportion of the records that were available in the node's cache or replied to the node in the first 10s after the broadcast of the query.

As Fig. 10.24 shows, even in adverse conditions, SIPCache is able to retrieve on average approximately 80% of the records satisfying the query. When the number of neighbors is on average 14 (corresponding to a transmission range

of 225m), results are already above 95%. It should be noted that these results are obtained with the propagation of the query to at most 2 hops away from the source of the query. Better results can be obtained by increasing the maximum TTL of the query at the expenses of an increasing number of messages. Not surprisingly, "Cache 100" is capable of, in the majority of the cases, returning all records satisfying the query that have been advertised.

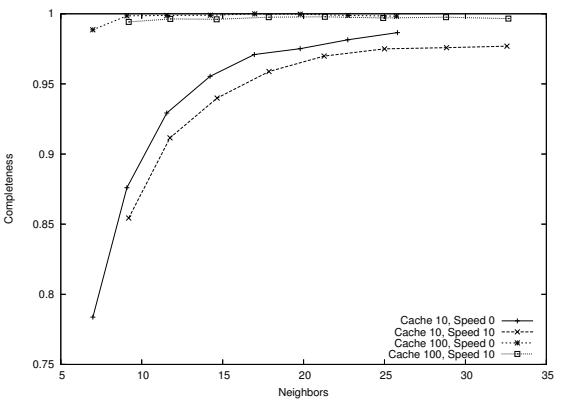


Figure 10.24: Coverage of Profile Based Queries

Figs. 10.25 and 10.26 evaluate the traffic generated in the network by the different operations. Results account with all retransmissions of every message. For queries, both the message dissemination and the replies have been accounted.

The figures show that the number of bytes required by profile based queries grows with the density of the network, contrary to the decreasing number of bytes required by targeted queries and by registrations. The justification is related with the implementation of PCache. The aim of the registration operation is to cover as much as possible all nodes in the network. Therefore, if nodes are closer to each other, less messages are required. On the other hand, profile based queries are bounded by a predefined TTL value. If nodes are closer, propagation of the query will cover an increasing number of nodes, which will produce an increasing number of replies.

The values presented are considered satisfactory for the application domain of SIPCache: the maximum observed value for a single operation is below 70Kbytes, which corresponds to less than 700 bytes per participant.

To evaluate the adequacy of SIPCache, we defined a simpler implementation of dSIP, which dismisses the registration/dissemination phase. In other words, this naïve implementation is equivalent to use pure broadcast in multi-hop networks as done in the link-local case. Because data are not replicated as a consequence of the dissemination, all queries are broadcasted to the network and only the nodes

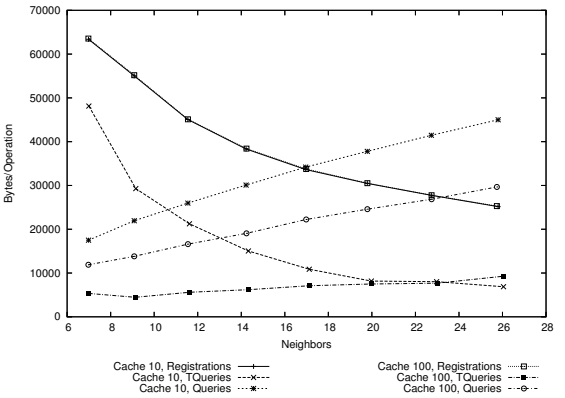


Figure 10.25: Bytes per operation in "Speed 0" tests

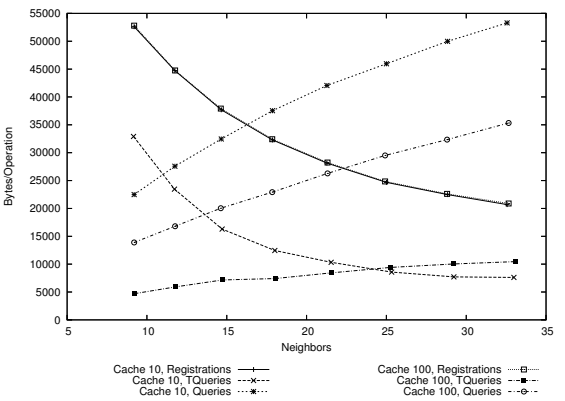


Figure 10.26: Bytes per operation in "Speed 10" tests

which have a binding satisfying the query send a reply. The broadcast operation required in this implementation is comparable with the registration operation defined in SIPCache if the overhead of sending a reply and the need to increase the probability of delivery of the messages to every node is neglected. Fig. 10.27 compares this naïve implementation with SIPCache for "Speed 0" tests. The y axis shows the gain (in number of messages per operation) of using SIPCache instead of the naïve implementation for the number of queries represented in the x axis. For a given number of queries, the figure shows how many excess messages are sent with SIPCache or the naïve implementation. Values above 0 in the message gain, imply that SIPCache saved Y number of messages when X number of queries are executed.

The results for this naïve implementation were estimated from multiplying the

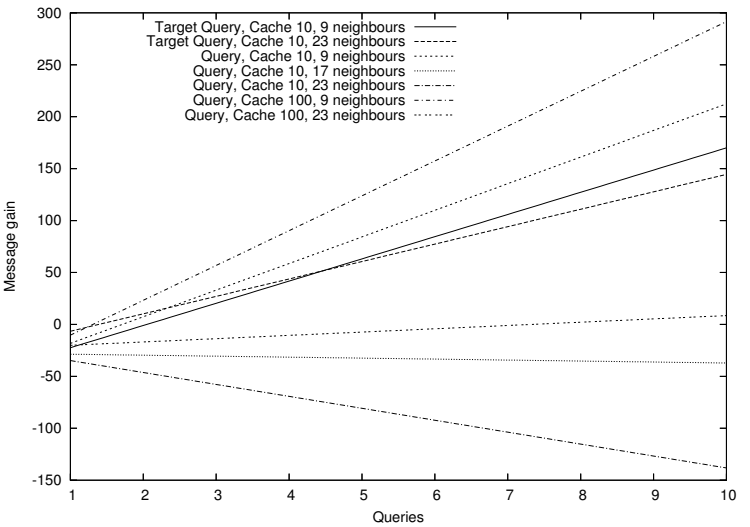


Figure 10.27: Comparison of SIPCache with a naïve implementation

average number of messages used in the registration operations of SIPCache by the number of queries. The cost of delivering replies was considered negligible as in this implementation it would consist in a point-to-point delivery for each node responding to the query. For the estimation in SIPCache, we have accounted with the original cost of the registration and with the average number of messages required for propagating the query and replies.

Results show that in the majority of the cases, SIPCache begins to save device resources after a small number of queries. We emphasize that in a real deployment, the number of queries should exceed by large the number of registered users, amplifying the expected savings of device' resources. Exceptions are profile based queries in "Cache 10" tests with a large density. This negative value can be attenuated by the gains presented by targeted queries in the same situation. It should also be noted that bigger caches, smaller proportions of the records satisfying the query or lower TTL values all would contribute to the increase of the efficiency of profile based queries.

Chapter 11

Conclusions and Future Work

The standard SIP protocol, in many cases, does not allow beginning sessions without the support of centralized entities, because the SIP end clients do not know the exact location of the party to contact, and rely on the SIP servers for discovering it. The key factor enabling decentralized SIP is embedding SIP registrar and proxy server functionalities in each end device, in order to make SIP operations truly decentralized.

11.1 Summary of Results

This dissertation has presented a solution for enabling a decentralized session management framework based on the Session Initiation Protocol. The framework has been mainly designed to fit the characteristics of a distributed wireless environment, such as ad-hoc networks, where bandwidth availability and scarce terminal resources are an issue. The presented solution, decentralized SIP, allows discovery of SIP users in an ad-hoc network and the establishment of SIP sessions in a completely distributed way and without support of centralized servers, belonging to the infrastructured network. Security support is provided, so that users can enforce the authenticity of the sender of a SIP message, and verify the integrity of a received message.

On top of the distributed session management functionalities for ad-hoc networks, the framework was extended in two other directions. First, interoperability with the Internet is ensured. A decentralized SIP device can be used transparently for the user and for any SIP application in the Internet, to communicate following standard SIP procedures with standard nodes in the Internet. Moreover, the framework enables communication between nodes in heterogeneous environments, allowing a user in the Internet to begin a session with a user in a non-isolated ad-hoc network, where one of the nodes provides Internet access. The second enhance-

ment was building applications that could benefit from the functionalities offered by the middleware framework. We built instant messaging and presence services based on SIP, enhancing them with security support and keeping the interoperability with the Internet. Finally, the second step was building a more advanced security management application, allowing users to exchange SIP security certificates while in the ad-hoc network, search for a remote user's certificate, or request a third party user's certificate from a trusted source.

The dissertation was divided into two main parts. The first part presented a background overview of technologies related to its subject. The SIP protocol and the extensions that are relevant for our purposes were given. The dissertation also presented an overview of possible target network environments where decentralized SIP can be deployed. To conclude the background part, several ways for distributing information in decentralized environments were illustrated.

The second part of the dissertation presented and discussed our solution in detail. Decentralized SIP is designed primarily for small-scaled wireless ad-hoc networks, and its operations optimized for such a network environment. Decentralized SIP allows not only to manage SIP sessions without the support of centralized servers, but also to discover the identity of users in ad-hoc networks. This characteristic is of fundamental importance as in ad-hoc networks knowledge on users in the vicinity is often previously unknown to a user. Without knowledge of the user name (and, of course, the contact IP address) of the users in the network, it is not possible to initiate sessions. Several flavors of decentralized SIP have been presented.

The main difference among these alternative versions of decentralized SIP lies in the way in which operations of user discovery are performed. We designed and implemented two solutions. Fully distributed SIP or dSIP utilizes SIP-only methods to perform user discovery. SLP-aided SIP or sSIP utilizes the Service Location Protocol, a service discovery protocol, instead of SIP for the same operations. Once the contact information of users in the ad-hoc network has been retrieved, SIP sessions can be initiated and the two methods coincide. Within dSIP, two modes of operations have been identified. They refer to whether the discovery of users present in the ad-hoc network is performed proactively, even though the user has expressed no explicit wish to begin a SIP session, or reactively. In this latter case, user discovery is done when a user decides to initiate a SIP session.

Further, we presented PCache, an algorithm for efficiently caching and distributing data items in a multi-hop ad-hoc network. The aim of this algorithm is to produce, by means of replication, a distribution of a data item in an ad-hoc network so that any node issuing a query for the item is able to retrieve from a node placed within a minimum number of hops from the querying node, ide-

ally within its first hop neighborood. The general algorithm was fine tuned and adapted for decentralized SIP to be used as the application generating the data items; we called the combination of decentralized SIP and PCache SIPCache. In this context, SIPCache was used to enable a distributed SIP location service for ad-hoc networks, and to spread a user's binding in the network.

The introduction of the cooperative caching policies and dissemination algorithms of SIPCache solves a limitation of the standard version of decentralized SIP in big ad-hoc networks. In principle, users need to store the bindings of all the other users in the network, and to reply with a unicast message to a received broadcast REGISTER message in order to exchange the binding with a registering user. In small networks, our analytical results show that the approach chosen for broadcast-based decentralized SIP registration is scalable; in big multi-hop ad-hoc networks enhancements are needed to maintain the efficiency. The design of storing the bindings of all the other users locally should be relaxed. The policy of replying to a received registration should be revised, as the route discovery traffic generated by replying to each and every registration message with a unicast reply would make the network unavailable for any user data flow. The distributed data spreading and retrieving mechanism of SIPCache addresses this problem efficiently, as the simulation results have shown.

An implementation of decentralized SIP was presented. The software modules of which decentralized SIP is comprised of were discussed. An example of message exchange was provided, followed by the presentation of experiments made with a testbed implementation. These experiments proved the effectiveness of decentralized SIP in achieving its design targets. The proposed solution is enhanced with security support. The main goal of the security mechanism for decentralized SIP is ensuring the integrity of exchanged signaling messages and the identity of the sender of a message. Adding confidentiality of signaling messages was not considered worthwhile, within the constraints of our environment, as the benefits it brings about are not worth the increased complexity and overhead. The scarce resource availability of terminals and wireless links were taken into account at the design phase.

The evaluation section analyzed the overhead of the various versions of decentralized SIP. Proactive dSIP was compared to reactive dSIP and dSIP compared to sSIP. The results of the analysis show that it is not possible to state that proactive dSIP is absolutely a better approach, compared to reactive dSIP or vice versa. The choice of which approach introduces lower bandwidth requirements strongly depends on the size of the network and other parameters, like the activity rate of the nodes in the network. However, the outcome of the analysis suggests that reactive dSIP can be more suitable in smaller networks, especially if used by applications that do not require continuous session establishments. In the case of the lecture

room, for example, the activity of nodes is very low as there is only one active media source. In this case reactive dSIP can outperform proactive dSIP. In other scenarios, like in an airport, where it is not possible to predict the rate of session initiation requests, proactive dSIP may be more bandwidth friendly than reactive dSIP. In general, proactive and reactive dSIP have advantages and disadvantages similar to those of proactive and reactive routing protocols, when they are compared.

A general advantage of proactive dSIP lies in the bindings availability. Since the discovery of users is done proactively, when a session initiation request is issued, the list of available users is already present in the local device. In reactive dSIP it would be retrieved at the moment implying a certain delay in session establishment. Some users may consider bindings availability a more desirable property than the amount of data sent on the network, also in consideration that in ad-hoc networks no payments are due to network operators for the exchanged data as would be the case if communication happens through the operator infrastructure (GPRS, IMS).

The results of the analysis on the overhead of dSIP and sSIP tell that, as expected, dSIP has more overhead than sSIP, since it uses large, text-based SIP messages for user discovery. On the other hand, SLP is binary encoded and its messages more compact in size. The overhead of dSIP is higher both when no security is requested and when messages are protected for authenticity. In this latter case, the dSIP Identity solution was shown to be more efficient, in terms of how much overhead it adds to the basic solution, than the SLP enhanced authentication block.

However, the choice of the better option between dSIP and sSIP cannot be based only on the bandwidth requirements. One advantage of dSIP over sSIP is the interoperability between different solutions. SIP functionalities are mandatory to implement if session-based applications are to be deployed; therefore, using SIP for user discovery operations as well is reasonable. On the other hand, in ad-hoc networks a service discovery protocol is often necessary to discover which services are available; even if dSIP is used, an ad-hoc node may need to use a service discovery protocol to acquire information about the availability of other, non-SIP, services in the network. If bandwidth limitations are an issue, sSIP can thus be chosen instead of dSIP for SIP user discovery.

Another point that favors the use of dSIP over sSIP is the availability of the bindings at each node. With dSIP, with a single message handshake the bindings are exchanged between the newcomer and the other nodes in the ad-hoc network; with SLP, this is not possible, as the bindings cannot be retrieved from an SLP query. In an SLP query, in fact, the AOR of the querying user is not passed to the other nodes. With sSIP, broadcast (or multicast) SLP requests for users in the

network can be periodically refreshed; this allows the broadcasting node to receive replies from newcomers. However, a node would have to wait for the expiration of its registration refresh timeout to discover the newcomers' identities.

If the user or local policies require the confidentiality security level for the user discovery phase, dSIP cannot be used because protecting dSIP messages for confidentiality using S/MIME would make SIP messages to big to be efficiently handled. On one hand, S/MIME messages would easily overcome that maximum allowed size for SIP over UDP, on the other hand, S/MIME can be too complex to implement and run on small handheld devices. Other existing security solutions were not applicable in the dSIP operating environment.

The solution that would ensure confidentiality at the price of reasonable bandwidth consumption is sSIP. In the implementation of SLP used for our experiments, user discovery happened in two phases; first a broadcast SLP Query for the service SIP was issued by the registering node, and after, a broadcast query for the service attribute AOR was sent. The scope of the first query was retrieving the IP address of the replying nodes, while the second query was sent to fetch the AOR of the SIP users in the network. In wireless ad-hoc networks where the medium is shared and collisions are an issue, it is preferable to reduce the number of messages sent over the air, even if this would come at the price of increasing their size. From this point of view, dSIP is better than sSIP, as it only requires a single phase of broadcast - unicast messages handshake.

In summary, the two alternative solutions can and should coexist, as one option has desirable features, which instead are missing in the other alternative. From a practical point of view, the user could be prompted to choose which user discovery mechanism he wants to use. The choice depends, among others, on the operational environment, on the application using the underlying services and on the security level requested.

The session management framework we have built is complete and flexible. It is complete in the sense that we have equipped it with a broad range of additional services on top of the main session management service. The framework can interact with a service discovery protocol to enforce the SIP location service, and it is enhanced with security support. Most importantly, it can transparently interoperate with standard SIP, both when a decentralized SIP node is used in ad-hoc networks to communicate with an Internet node, and when the decentralized SIP device is used in the Internet natively. On top of middleware services, we have built a proof-of-concepts application suite, which leverages instant messaging and presence services in ad-hoc and heterogeneous networks and advanced security support.

11.2 Future Work

This dissertation is only one step towards a truly ubiquitous deployment of Internet services, but it surely is not enough alone. There is still much work to do to make the Internet truly mobile, as conveniently for the user as cellular telephony. We plan to give our contributions to fulfill this vision by extending and improving the work presented in this dissertation.

One of the first development directions consists in enhancing the interactions with the service discovery framework. We have seen that using SLP for user discovery can be useful, and more convenient than using SIP, e.g., when bandwidth availability is an issue. In order to fully utilize the capabilities of the SESSI service discovery framework, we plan to extend the SIP local server capabilities to support passive service discovery. In this way, decentralized SIP users can advertise their bindings by means of SLP messages, in a fashion similar to spreading one own REGISTER to the ad-hoc network.

A possible direction for improving security support is to study how Datagram TLS can be applied to SIP, and hence decentralized SIP. If DTLS can be efficiently used as transport for SIP, confidentiality of decentralized SIP exchange can be obtained at lower layers, respecting the requirements of low bandwidth consumption and solution complexity. We will also closely follow the work done in the Better Than Nothing Security (BTNS) IETF working group to study how self-signed certificates can be used to establish IPSec tunnels in our frameworks. This concept has already been utilized in the SESSI framework to allow IPSec-protected connections to the gateway. Running decentralized SIP over DTLS provides confidentiality to the signaling exchange; the framework can also be extended to ensure the privacy of the users. One starting point for this issue is providing a hash of the AOR (and possibly of the Contact header field) instead of the sender's user name in plain text. The hash would be signed with the sender's certificate, and only users who have such a certificate can verify the identity of the sender. This method would extend and not substitute the certificate management framework already provided by decentralized SIP.

The PCache algorithm will be extensively studied and enhanced. PCache uses several parameters, and there is still lots of space for improvement by carefully studying the combination of parameters that make the algorithm better, while taking into consideration the network conditions, such as topology, node mobility, size and so on. Comparison of PCache with other data distribution and cache management algorithms will be useful to find new points of improvement of the algorithm. It will also be interesting to study how PCache interacts with the other components of the SESSI framework, such as for distributing SLP messages in big multi-hop ad-hoc networks. SLP messages are in fact much smaller than SIP messages, and therefore, PCache would better adapt when used with smaller mes-

sages as carriers, as this allows more items to be carried within a single item, and thus more shuffling and better data distribution. The most difficult part is in mapping PCache messages to SLP messages: SLP, also because its messages are binary encoded, is harder to adapt and extend than the SIP protocol, which is indeed designed with flexibility and extendability in mind.

SIPCache can be improved as well; particularly, it is interesting to evaluate how the message body format affects the performance of the algorithm. This aspect is important especially when profile-based queries are issued, but in general, in any type of SIPCache message. Since a higher number of complementary items leverages a better distribution of data items in the network, it is reasonable to expect that a binary encoding of SIPCache items will lead to increased performance, as opposed to plain text or XML-encoding. It must however be analyzed whether the encoding/decoding process of binary data causes penalties from processing point of view, as compared to plain text parsing, or XML-parsing. A further improvement direction for SIPCache is to study how malicious nodes can impact its performance, when bogus users contact information is spread through disseminated messages. Selfish behavior, where SIPCache nodes dump all the received messages without retransmitting them will also be subject to investigation.

We have described how SIPCache messages can be protected for integrity using the SIP Identity mechanism. Our solution fits well into the decentralized SIP framework and respects the operating principles of SIP Identity, but requires that a chain of trust is established among all the nodes that forward a SIPCache message. It is interesting to study how to break this chain of trust. One way is to let each node forwarding a message sign only the part of body it has modified. In this way, users can verify only parts of the message, breaking the all-or-nothing chain of trust. This approach has the problem that adding several signatures, each 240 bytes long according to the SIP Identity specification, would make the SIP Identity method inefficient. The SIP Identity mechanism advantage lies in the low overhead with which it ensures integrity protection; adding several signatures would nullify this advantage. A partial solution to this problem would be to use a different and less powerful signing mechanism, so that signatures would be less than 240 bytes. This involves changes into the way SIP Identity is used, possibly defining new header fields to specify the type of hashing used in each signature, so it is not a straightfoward solution, although worthy of future research.

However, we believe that the most important step to take in order to achieve our goal of making the Internet ubiquitous is to develop a broad set of applications, which are appealing for the end user. Eventually, it is the availability of useful and interesting applications that will push the mainstream interest for mobile Internet upwards. The underlying session management framework, and supporting middleware in general, is a necessary enabler that needs to be present, but

in itself, it cannot push the Internet to become truly mobile. As part of the functional enabler middleware, it is crucial to include the interoperability with standard Internet applications in the two use cases of Internet communication and heterogeneous communication. As part of this research item, in order to improve the visibility of this application towards the masses, it would be interesting to standardize the modifications to SIP operations brought about to execute advanced certificate management service with dSIP. We have already built an instant messaging and presence application, which interacts with our session management framework and is based on standard IETF protocols. This allows interoperability with Internet applications. The MSRP relay allows the application to be deployed in heterogeneous environments. There are still two main directions where research on application side can be done, and they both point towards enabling complete real time communication capabilities in mobile Internet. VoIP audio/video calls and group communication are the application families that would probably push the concept of mobile Internet.

So far, we have enabled point-to-point communication. However, most of the use cases envisaged for decentralized SIP have a fundamental component in group communication. For example, in a lecture room or a meeting, there is the need for a one-to-many transfer of media, and therefore one-versus-many session establishment procedures. In fact, the media source should not necessarily invite recipients to the sessions, but rather it could advertise the media flow and participants may join if they wish. In the SIPPING WG there is work ongoing, several internet drafts and RFCs, for defining a framework for SIP conferencing. Group communication is a broad subject, as it can refer to conference calls, chat rooms, or even on-line multi-player gaming.

There are two important factors to study in order to enable group communication and VoIP audio/video calls: one is the actual deployment of the applications in a complete peer-to-peer fashion, and the other is studying how and whether the ad-hoc networking environment can support applications with strict timing requirements, especially when multi-hop communication is involved. Studies on routing and lower layers are needed on this aspect.

There are several obstacles ahead on the road to a ubiquitous Internet: this dissertation has shown solutions for some of them. When all the obstacles will be removed, the vision of mobile Internet, anywhere and anytime will become closer, and users will be able to enjoy the commodity of Internet on the move, as easily and smoothly as they currently do with cellular phones. And yes, this is our ultimate goal, and desire, that Internet moves from its fixed paradigm to a more mobilized nature, similarly to what happened with cellular and traditional fixed-line telephony. We believe that this is not only a dream, but a realistic target, and we will continue to give our contribution to make it become true.

References

[1] Third Generation Partnership Project 3GPP. IP multimedia subsystem (IMS); stage 2. Technical Report 3GPP TS 23.228 version 6.9.0, March 2005.

[2] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Elsevier Journal on Ad-Hoc Networks*, 2(1):1–22, January 2004.

[3] P. G. Argyroudis et al. Performance analysis of cryptographic protocols on handheld devices. In *The Third IEEE International Symposium on Network Computing and Applications (NCA'04)*, 2004.

[4] D. Atkins, W. Stallings, and P. Zimmermann. PGP Message Exchange Formats. RFC 1991 (Informational), August 1996.

[5] N. T. J. Bailey. *The Mathematical Theory, of Infectious Diseases and its Applications*. Hafner Press, 2 edition, 1975.

[6] R. Baldoni, G. Cortese, F. Davide, and A. Melpignano ed. *Epidemic Dissemination for Probabilistic Data Storage, chapter of Global Data Management*. IOS, July 2006.

[7] N. Banerjee, A. Acharya, and S. K. Das. Peer-to-peer sip-based services over wireless ad hoc networks. In *BROADWIM: Broadband Wireless Multimedia Workshop*, October 2004.

[8] S. Baset, H. Schulzrinne, E. Shim, and K. Dhara. Requirements for SIP-based peer-to-peer internet telephony. Internet-draft (work in progress), IETF, October 2005. draft-baset-sipping-p2preq-00.

[9] S. A. Baset and H. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. Technical report number: CUCS-039-04, Columbia University, New York, September 2004.

[10] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, jul–sep 2003.

[11] C. Boulton, J. Rosenberg, and G. Camarillo. Best current practices for nat traversal for SIP. Internet draft (work in progress), IETF, June 2006. draft-ietf-sipping-nat-scenarios-05.

[12] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reservation protocol (RSVP), version 1 functional specification. Request for Comments (Standards Track) 2205, Internet Engineering Task Force, September 1997.

[13] D. Bryan and C. Jennings. A P2P approach to SIP registration. Internet-draft (work in progress), IETF, October 2006. draft-bryan-sipping-p2p-03.

[14] D. Bryan, E. Shim, and B. Lovekamp. Use cases for peer-to-peer session initiation protocol (P2P SIP). Internet-draft (work in progress), IETF, November 2005. draft-bryan-sipping-p2p-usecases-00.

[15] C. Jennings C., R. Mahy, and A. B. Roach. Relay extensions for message sessions relay protocol (MSRP). Internet draft (work in progress), IETF, December 2006. draft-ietf-simple-msrp-relays-10.

[16] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428 (Proposed Standard), December 2002.

[17] R. Chandra, V. Ramasubramanian, and K. Birman. Anonymous gossip: improving multicast reliability in mobile ad-hoc networks. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 275–283, April 2001.

[18] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005.

[19] S. Cheshire and M. Krochmal. Multicast DNS. Internet draft (work in progress), IETF, August 2006. draft-cheshire-dnsext-multicastdns-06.

[20] S. Ci and H. Sharif. A link adaptation scheme for improving throughput in the IEEE 802.11 wireless LAN. In *Proc. 27th Annual IEEE Conf. on Local Computer Networks (LCN 2002)*, pages 205–208, 2002.

[21] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[22] Microsoft Corporation. *Web Services Dynamic Discovery (WS-Discovery)*, April 2005. Available at `http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Discovery.pd%f`, October 9, 2007.

[23] A. Datta, S. Quarteroni, and K. Aberer. Autonomous gossiping: A self-organizing epidemic algorithm for selective information dissemination in wireless mobile ad-hoc networks. In *Proceedings of the International Conference on Semantics of a Networked World ICSNW 2004*, pages 126–143, 2004.

[24] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, 1987.

[25] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Updated by RFC 3546.

[26] B. Campbell ed., R. Mahy ed., and C. Jennings ed. The message session relay protocol (MSRP). Internet draft (work in progress), IETF, December 2006. (draft-ietf-simple-message-sessions-18).

[27] C. Jennings ed. and A. Hawrylyshen. SIP conventions for UAs with outbound only connections. Internet-draft (work in progress), IETF, February 2005. draft-jennings-sipping-outbound-01.

[28] P. Engelstad, D.V. Thanh, and G. Egeland. Name resolution in on-demand MANETs and over external ip network. In *Proceedings of the IEEE International Conference on Communications. ICC '03*, volume 2, pages 1024–1032, May 2003.

[29] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60–67, May 2004.

[30] Z. Fan and S. Subramani. An address autoconfiguration protocol for IPv6 hosts in a mobile ad hoc network. *Elsevier Journal on Computer Communications*, 56(4):339–350, March 2005.

[31] J. Favela, C. Navarro, and M. Rodriguez. Extending instant messaging to support spontaneous interactions in ad-hoc networks. In *CSCW '02: Conference on Computer Supported Cooperative Work, New Orleans, USA*, 2002.

[32] L. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. of the 20th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2001)*, volume 3, pages 1548–1557, 2001.

[33] E. Ferro and F. Potorti. Bluetooth and IEEE 802.11 wireless protocols: A survey and comparison. *IEEE Wireless Communications Magazine*, 12(1):12–26, February 2005.

[34] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. Request for Comments 2616 (Standard), Internet Engineering Task Force, June 1999.

[35] UPnP Forum. Upnp forum website. At `http://www.upnp.org/`, May 2005.

[36] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication. RFC 2617 (Draft Standard), June 1999.

[37] D. Gavidia, S. Voulgaris, and M. Van Steen. Epidemic-style monitoring in large-scale wireless sensor networks. Technical report number: IR-CS-012.05, Vrije Universiteit, Amsterdam, the Netherlands, March 2005.

[38] K. Geihs. Middleware challenges ahead. *IEEE Computer*, 34(6):24–31, June 2001.

[39] D. Greene and D. O'Mahony. Instant messaging & presence management in mobile ad-hoc networks. In *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, Orlando, USA*, 2004.

[40] Wireless USB Promoter Group. Wireless USB: The first high-speed personal wireless interconnect. White paper, February 2004. Available at: `http://www.usb.org/developers/wusb/docs/wirelessUSB.pdf`, accessed October 9, 2007.

[41] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2 . RFC 2608 (Proposed Standard), June 1999. Updated by RFC 3224.

[42] Z. J. Haas, J.Y. Halpern, and L. Li. Gossip-based ad hoc routing. In *INFOCOM '02: Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 23–27, June 2002.

[43] M. Handley and Jacobson V. (SDP): Session description protocol. Request for Comments 2327 (Standards Track), Internet Engineering Task Force, April 1998.

[44] S. Helal. Standards For Service Discovery And Delivery. *IEEE Pervasive Computing*, 1(3):95–100, July-September 2002.

[45] T. Hou, L. Tsao, and H. Liu. Analyzing the throughput of IEEE 802.11 DCF scheme with hidden nodes. In *Proc. of the IEEE 58th Vehicular Technology Conference (VTC 2003-Fall)*, volume 5, pages 2870– 2874, 2003.

[46] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 146–155, 2001.

[47] International Telecommunication Union ITU. Packet-based multimedia communications systems. recommendation h323 version 5. Technical report, July 2003.

[48] C. Jennings and N. Modadugu. Using DTLS as a transport for SIP. Internet draft (work in progress), IETF, July 2005. draft-jennings-sip-dtls-01.

[49] C. Jennings, J. Peterson, and J. Fischl. Certificate management service for SIP. Internet-draft (work in progress), IETF, October 2006. draft-ietf-sipping-certs-02.

[50] J. Jeong, J. Park, and N. Kim. DNS name service based on secure multicast DNS for IP-v6 mobile ad hoc networks. In *Proceedings of the 6th International Conference on Advanced Communication Technology*, volume 1, pages 3–7, 2004.

[51] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[52] D. B. Johnson, D. A. Maltz, and Y. Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet draft (work in progress), IETF, July 2004. draft-ietf-manet-dsr-10.txt.

[53] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, and K. Summers. Session initiation protocol (SIP) basic call flow examples. Request for Comments 3665 (Best Current Practice), Internet Engineering Task Force, December 2003.

[54] A. Johnston and H. Sinnreich. SIP, P2P, and internet communications. Internet-draft (work in progress), IETF, March 2006. draft-johnston-sipping-p2p-ipcom-02.

[55] A. Johnston, R. Sparks, C. Cunningham, S. Donovan, and K. Summers. Session initiation protocol service examples. Internet draft (work in progress), IETF, January 2007. draft-ietf-sipping-service-examples-12.

[56] L. Källström, S. Leggio, J. Manner, T. Mikkonen, K. Raatikainen, J. Saarinen, S. Suoranta, and A. Ylä-Jääski. A framework for seamless service interworking in ad-hoc networks. *Elsevier Journal on Computer Communications*, 29:3277–3294, October 2006.

[57] L. Källström and J. Saarinen. Secure service discovery protocol implementation for wireless ad-hoc networks. In *1st International Wireless Summit, Aalborg, 17-22 September*, 2005.

[58] R. H. Katz. Adaptation and mobility in wireless information systems. *IEEE Communications Magazine*, 40(2):102–114, May 2002.

[59] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401 (Proposed Standard), November 1998. Updated by RFC 3168.

[60] J. Kettunen. *Decentralized Session Initiation Protocol Framework in Multihop Ad hoc Routing Environment*. Msc. thesis, Department of Computer Science and Engineering, Helsinki University of Technology, Finland, May 2006.

[61] H. Khlifi, A. Agarwal, and J.C. Gregoire. A framework to use SIP in ad-hoc networks. In *Canadian Conference on Electrical and Computer Engineering. IEEE CCECE*, volume 2, pages 985–988, May 2003.

[62] L. Kleinrock. Breaking loose. *Communications of the ACM*, 44(9):41–46, September 2001.

[63] T. Kunz and E. Cheng. On-demand multicasting in ad-hoc networks: Comparing AODV and ODMRP. In *ICDCS 2002: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 453–454, July 2002.

[64] J. Kurian. Design and implementation of an authentication and authorization module for service access in ad hoc networks. Master's thesis, Helsinki University of Technology, February 2005. 77 pp.

[65] S. Lee, M. Gerla, and C. Chiang. On-demand multicast routing protocol. In *WCNC 1999: Proceedings of the 1999 IEEE Wireless Communications and Networking Conference*, pages 21–24, September 1999.

[66] S. Leggio, S. Liimatainen, J. Manner, T. Mikkonen, J. Saarinen, and A. Ylä-Jääski. Towards service interworking among ad-hoc networks and the internet. In *14th IST Mobile and Wireless Communications Summit, Dresden, 19-23 June*, 2005.

[67] S. Leggio, J. Manner, A. Hulkkonen, and K. Raatikainen. Session initiation protocol deployment in ad-hoc networks: a decentralized approach. In *Proceedings of the Second International Workshop on Wireless Ad-Hoc Networks, IWWAN'05*, London, May 2005.

[68] S. Leggio, J. Manner, and K. Raatikainen. A secure SIP-based instant messaging and presence framework for ad-hoc networks. In *To appear in the Symposium on Wireless Communications and Networking of 2006 IEEE GLOBECOM Conference, 27 November - 1 December 2006, San Francisco, CA, USA*.

[69] S. Leggio, H. Miranda, K. Raatikainen, and L. Rodrigues. SIPCache: A distributed SIP location service for mobile ad-hoc networks. In *The 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS 2006) July 17-21, 2006 - San Jose, California*.

[70] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proc. of the 1st USENIX/ACM Symp. on Networked Systems Design and Implementation (NSDI 2004)*, 2004.

[71] S. Ludwig, J. Beda, P. Saint-Andre, J. Hildebrand, S. Egan, and R. McQueen. Jingle. Experimental, Jabber Software Foundation, December 2006. XEP-0166.

[72] S. Ludwig, P. Saint-Andre, and S. Egan. Jingle audio content description format. Experimental, Jabber Software Foundation, December 2006. XEP-0167.

[73] J. Luo, P. T. Eugster, and J.-P. Hubaux. Route driven gossip: probabilistic reliable multicast in ad hoc networks. In *INFOCOM '03: Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 2229–2239, March/April 2003.

[74] Handley M., Schulzrinne H., Schooler E., and Rosenberg J. SIP: Session initiation protocol. RFC 2543 (Standards Track), IETF, March 1999.

[75] Lonnfors M., Leppanen E., Khartabil H., and Urpalainen J. Presence information data format (PIDF) extension for partial presence. Internet draft (work in progress), IETF, November 2006. (draft-ietf-simple-partial-pidf-format-08).

[76] Lonnfors M., Costa-Requena J., Leppanen E., and Khartabil H. Session initiation protocol (SIP) extension for partial notification of presence information. Internet draft (work in progress), IETF, July 2006. (draft-ietf-simple-partial-notify-08).

[77] J. Manner, S. Leggio, and K. Raatikainen. An internet SIP gateway for ad-hoc networks. In *Proceedings of the IEEE Sensor and Ad Hoc Communications and Networks (SECON)*, volume 3, pages 786–791, June 2006.

[78] J. Manner, A. Ylä-Jääski, and J. Kettunen. Efficient distributed service location and session management for ad-hoc networks. In *Proceedings of WOWMOM 2007: 8th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*.

[79] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen. An algorithm for dissemination and retrieval of information in wireless ad hoc networks. In *Proceedings of EUROPAR 2007*.

[80] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen. An algorithm for distributing and retrieving information in sensor networks. In *Brief announcement track in OPODIS 2006: 10th International Conference On Principles Of Distributed Systems*.

[81] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen. A power-aware broadcasting algorithm. In *17th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications PIMRC 2006 11-14 September, Helsinki, Finland*.

[82] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen. A stateless neighbour-aware cooperative caching protocol for ad-hoc networks. DI/FCUL TR 05–23, Department of Informatics, University of Lisbon, December 2005. Also as Technical Report Number C–2005–76. Computer Science Department, University of Helsinki.

[83] A. Misra, S. Das, A. McAuley, and S. K. Das. Autoconfiguration, registration, and mobility management for pervasive computing. *IEEE Personal Communications Magazine*, 8(4):24–31, August 2001.

[84] N. Modadugu and E. Rescorla. The design and implementation of datagram tls. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA*.

[85] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461 (Draft Standard), December 1998.

[86] S. Nesargi and R. Prakash. MANETconf: configuration of hosts in a mobile ad hoc network. In *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE INFOCOM*, volume 2, pages 1059 – 1068, June 2002.

[87] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, 1999.

[88] A. Niemi. Session Initiation Protocol (SIP) Extension for Event State Publication. RFC 3903 (Proposed Standard), October 2004.

[89] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). RFC 3684 (Experimental), February 2004.

[90] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[91] C.E. Perkins, J. T. Malinen, R. Wakikawa, and Y. Sun. IP address autoconfiguration for ad hoc networks. Internet-draft (work in progress), IETF, January 2001. draft-ietf-manet-autoconf-01.txt.

[92] J. Peterson. A privacy mechanism for the Session Initiation Protocol. RFC 3323 (Standards Track), IETF, November 2002.

[93] J. Peterson and C. Jennings. Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP). RFC 4474 (Proposed Standard), August 2006.

[94] H. Pucha, S. M. Das, and Y. C. Hu. Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks. In *WMCSA '04: Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, pages 163–173, December 2004.

[95] K. Raatikainen. A new look at mobile computing. In *Proccedings of ANWIRE workshop, Athens*, May 14th 2004. Available at `http://www.cs.helsinki.fi/u/kraatika/Papers/RaatikainenAnwireMay04.pdf`.

[96] B. Ramsdell. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. RFC 3851 (Proposed Standard), July 2004.

[97] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, ACM SIGCOMM*, volume 31, August 2001.

[98] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), April 2006.

[99] A. B. Roach. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265 (Proposed Standard), June 2002.

[100] J. Rosenberg. A Presence Event Package for the Session Initiation Protocol (SIP). RFC 3856 (Proposed Standard), August 2004.

[101] J. Rosenberg. A Framework for Conferencing with the Session Initiation Protocol (SIP). RFC 4353 (Informational), February 2006.

[102] J. Rosenberg. Obtaining and using globally routable user agent (UA) URIs (GRUU) in the session initiation protocol (SIP). Internet-draft (work in progress), IETF, October 2006. draft-ietf-sip-gruu-11.

[103] J. Rosenberg and H. Schulzrinne. An Offer/Answer Model with Session Description Protocol (SDP). RFC 3264 (Proposed Standard), June 2002.

[104] J. Rosenberg and H. Schulzrinne. Session Initiation Protocol (SIP): Locating SIP Servers. RFC 3263 (Proposed Standard), June 2002.

[105] J. Rosenberg and H. Schulzrinne. An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing. RFC 3581 (Proposed Standard), August 2003.

[106] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853.

[107] J. Rosenberg, H. Schulzrinne, and O. Levin. A Session Initiation Protocol (SIP) Event Package for Conference State. RFC 4575 (Proposed Standard), August 2006.

[108] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *MobiCom 1999: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 207–218, 1999.

[109] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920 (Proposed Standard), October 2004.

[110] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. RFC 3921 (Proposed Standard), October 2004.

[111] P. Saint-Andre. Link-local messaging. Experimental, Jabber Software Foundation, December 2006. XEP-0174.

[112] P. Saint-Andre and M. Chen. Jingle video content description format. Experimental, Jabber Software Foundation, December 2006. XEP-0180.

[113] S. Salsano, L. Veltri, and D. Papalilo. SIP security issues: the SIP authentication procedure and its processing load. *IEEE Network*, 16(6):38–44, November-December 2002.

[114] M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10–17, August 2001.

[115] H. Schulzrinne and J. Rosenberg. Internet telephony: architecture and protocols - an IETF perspective. *Elsevier Computer Networks*, 31(3):237–255, February 1999.

[116] H. Schulzrinne and J. Rosenberg. The session initiation protocol: Internet-centric signaling. *IEEE Communications Magazine*, Volume 38(N. 10), October 2000.

[117] M. Sheng, J. Li, and Y. Shi. Relative degree adaptive flooding broadcast algorithm for ad hoc networks. *IEEE Transactions on Broadcasting*, 51(2):216 – 222, June 2005.

[118] E. Shim, S. Narayanan, and G. Daley. An architecture for peer-to-peer session initiation protocol (P2P SIP). Internet-draft (work in progress), IETF, February 2006. draft-shim-sipping-p2p-arch-00.

[119] K. Singh and H. Schulzrinne. Peer-to-peer internet telephony using SIP. Technical report cucs-044-04, Columbia University, October 2004.

[120] R. Sparks. The Session Initiation Protocol (SIP) Refer Method. RFC 3515 (Proposed Standard), April 2003.

[121] R. Sparks, A. Johnston, and D. Petrie. Session initiation protocol call control - transfer. Internet draft (work in progress), IETF, October 2006. draft-ietf-sipping-cc-transfer-07.

[122] F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, 2000.

[123] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, ACM SIGCOMM*, volume 31, August 2001.

[124] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson. Presence Information Data Format (PIDF). RFC 3863 (Proposed Standard), August 2004.

[125] Sun Microsystems. *Jini Architecture Specification 2.0*, June 2003. Available at `http://wwws.sun.com/software/jini/specs/jini2_0.pdf`.

[126] Sun Microsystems. *Jini Technology Core Platform Specification 2.0*, June 2003. Available at `http://wwws.sun.com/software/jini/specs/core2_0.pdf`, October 9, 2007.

[127] R. Tafazolli ed. *Technologies for the Wireless Future. Wireless World Research Forum WWRF*. Wiley, 2005.

[128] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 2462 (Draft Standard), December 1998.

[129] C. Tschudin, P. Gunningberg, H. Lundgren, and E. Nordström. Lessons from experimental MANET research. *Elsevier Journal on Ad-Hoc Networks*, Volume 3(Number 3):221–233, March 2005.

[130] T. Ueda, S. Bandyopadhyay, and K. Hasuike. Implementing messaging services on ad-hoc community networks using proxy nodes. In *ADHOC '02: 2nd Swedish Workshop on Wireless Ad-Hoc Networks, Stockholm, Sweden*, 2002.

[131] U. Varshney and R. Jain. Issues in emerging 4g wireless networks. *IEEE Computer Journal*, 34(6):94–96, June 2001.

[132] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. Service Location Protocol. RFC 2165 (Proposed Standard), June 1997. Updated by RFCs 2608, 2609.

[133] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). RFC 2845 (Proposed Standard), May 2000. Updated by RFC 3645.

[134] W. Wang, S. Chang Liew, and V. O. K. Li. Solutions to performance problems in VoIP over a 802.11 wireless LAN. *IEEE Transactions on Vehicular Technology*, 54(1):366–384, January 2005.

[135] M. Weiser. The computer for the 21st century. *Scientific American*, 1991.

[136] K. Weniger and M. Zitterbart. Ipv6 stateless address autoconfiguration in large mobile ad hoc networks. In *Proceedings of the European Wireless 2002*, February 2002.

[137] B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 194–205, 2002.

[138] D. Willis and B. Hoeneisen. Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts. RFC 3327 (Proposed Standard), December 2002.

[139] J. Xi and C. Bettstetter. Wireless multi-hop Internet access: Gateway discovery, routing, and addressing. In *Proceedings of the International Conference on Third Generation Wireless and Beyond (3Gwireless)*, May 2002.

[140] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, November-December 1999.

[141] D. Zhu and M. Mutka. Sharing presence information and message notification in an ad hoc network. In *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, Dallas, USA*, 2003.

[142] The Third Generation Partnership Project. At `http://www.3gpp.org`, February 2005.

[143] The Internet Engineering Task Force IETF. At `http://www.ietf.org/`, February 2005.

[144] The International Telecommunication Union ITU-T. At `http://www.itu.int/ITU-T/`, February 2005.

TIETOJENKÄSITTELYTIETEEN LAITOS
PL 68 (Gustaf Hällströmin katu 2 b)
00014 Helsingin yliopisto

DEPARTMENT OF COMPUTER SCIENCE
P.O. Box 68 (Gustaf Hällströmin katu 2 b)
FIN-00014 University of Helsinki, FINLAND

JULKAISUSARJA **A**

SERIES OF PUBLICATIONS **A**

Reports may be ordered from: Kumpula Science Library, P.O. Box 64, FIN-00014 University of Helsinki, FINLAND.

A-1998-2 L. Kutvonen: Trading services in open distributed environments. 231 + 6 pp. (Ph.D. thesis).

A-1998-3 E. Sutinen: Approximate pattern matching with the q-gram family. 116 pp. (Ph.D. thesis).

A-1999-1 M. Klemettinen: A knowledge discovery methodology for telecommunication network alarm databases. 137 pp. (Ph.D. thesis).

A-1999-2 J. Puustjärvi: Transactional workflows. 104 pp. (Ph.D. thesis).

A-1999-3 G. Lindén & E. Ukkonen (eds.): Department of Computer Science: annual report 1998. 55 pp.

A-1999-4 J. Kärkkäinen: Repetition-based text indexes. 106 pp. (Ph.D. thesis).

A-2000-1 P. Moen: Attribute, event sequence, and event type similarity notions for data mining. 190+9 pp. (Ph.D. thesis).

A-2000-2 B. Heikkinen: Generalization of document structures and document assembly. 179 pp. (Ph.D. thesis).

A-2000-3 P. Kähkipuro: Performance modeling framework for CORBA based distributed systems. 151+15 pp. (Ph.D. thesis).

A-2000-4 K. Lemström: String matching techniques for music retrieval. 56+56 pp. (Ph.D.Thesis).

A-2000-5 T. Karvi: Partially defined Lotos specifications and their refinement relations. 157 pp. (Ph.D.Thesis).

A-2001-1 J. Rousu: Efficient range partitioning in classification learning. 68+74 pp. (Ph.D. thesis)

A-2001-2 M. Salmenkivi: Computational methods for intensity models. 145 pp. (Ph.D. thesis)

A-2001-3 K. Fredriksson: Rotation invariant template matching. 138 pp. (Ph.D. thesis)

A-2002-1 A.-P. Tuovinen: Object-oriented engineering of visual languages. 185 pp. (Ph.D. thesis)

A-2002-2 V. Ollikainen: Simulation techniques for disease gene localization in isolated populations. 149+5 pp. (Ph.D. thesis)

A-2002-3 J. Vilo: Discovery from biosequences. 149 pp. (Ph.D. thesis)

A-2003-1 J. Lindström: Optimistic concurrency control methods for real-time database systems. 111 pp. (Ph.D. thesis)

A-2003-2 H. Helin: Supporting nomadic agent-based applications in the FIPA agent architecture. 200+17 pp. (Ph.D. thesis)

A-2003-3 S. Campadello: Middleware infrastructure for distributed mobile applications. 164 pp. (Ph.D. thesis)

A-2003-4 J. Taina: Design and analysis of a distributed database architecture for IN/GSM data. 130 pp. (Ph.D. thesis)

A-2003-5 J. Kurhila: Considering individual differences in computer-supported special and elementary education. 135 pp. (Ph.D. thesis)

A-2003-6 V. Mäkinen: Parameterized approximate string matching and local-similarity-based point-pattern matching. 144 pp. (Ph.D. thesis)

A-2003-7 M. Luukkainen: A process algebraic reduction strategy for automata theoretic verification of untimed and timed concurrent systems. 141 pp. (Ph.D. thesis)

A-2003-8 J. Manner: Provision of quality of service in IP-based mobile access networks. 191 pp. (Ph.D. thesis)

A-2004-1 M. Koivisto: Sum-product algorithms for the analysis of genetic risks. 155 pp. (Ph.D. thesis)

A-2004-2 A. Gurtov: Efficient data transport in wireless overlay networks. [B 141 pp. (Ph.D. thesis)

A-2004-3 K. Vasko: Computational methods and models for paleoecology. 176 pp. (Ph.D. thesis)

A-2004-4 P. Sevon: Algorithms for Association-Based Gene Mapping. 101 pp. (Ph.D. thesis)

A-2004-5 J. Viljamaa: Applying Formal Concept Analysis to Extract Framework Reuse Interface Specifications from Source Code. 206 pp. (Ph.D. thesis)

A-2004-6 J. Ravantti: Computational Methods for Reconstructing Macromolecular Complexes from Cryo-Electron Microscopy Images. 100 pp. (Ph.D. thesis)

A-2004-7 M. Kääriäinen: Learning Small Trees and Graphs that Generalize. 45+49 pp. (Ph.D. thesis)

A-2004-8 T. Kivioja: Computational Tools for a Novel Transcriptional Profiling Method. 98 pp. (Ph.D. thesis)

A-2004-9 H. Tamm: On Minimality and Size Reduction of One-Tape and Multitape Finite Automata. 80 pp. (Ph.D. thesis)

A-2005-1 T. Mielikäinen: Summarization Techniques for Pattern Collections in Data Mining. 201 pp. (Ph.D. thesis)

A-2005-2 A. Doucet: Advanced Document Description, a Sequential Approach. 161 pp. (Ph.D. thesis)

A-2006-1 A. Viljamaa: Specifying Reuse Interfaces for Task-Oriented Framework Specialization. 285 pp. (Ph.D. thesis)

A-2006-2 S. Tarkoma: Efficient Content-based Routing, Mobility-aware Topologies, and Temporal Subspace Matching. 198 pp. (Ph.D. thesis)

A-2006-3 M. Lehtonen: Indexing Heterogeneous XML for Full-Text Search. 185+3 pp.(Ph.D. thesis).

A-2006-4 A. Rantanen: Algorithms for ^{13}C Metabolic Flux Analysis. 92+73 pp.(Ph.D. thesis).

A-2006-5 E. Terzi: Problems and Algorithms for Sequence Segmentations. 141 pp. (Ph.D. Thesis).

A-2007-1 P. Sarolahti: TCP Performance in Heterogeneous Wireless Networks.(Ph.D. Thesis).

A-2007-2 M. Raento: TCP Exploring privacy for ubiquitous computing: Tools, methods and experiments. (Ph.D. thesis).

A-2007-3 L. Aunimo: Methods for Answer Extraction in Textual Question Answering 127+18 pp. (Ph.D. Thesis).

A-2007-4 T. Roos: Statistical and Information-Theoretic Methods for Data Analysis. 82+75pp. (Ph.D. Thesis). for Heterogeneous Ad-Hoc and Fixed Networks 230 pp. (Ph.D. Thesis).