Year : 2013

System-level Support for Mobile Ad hoc Communications : an Algorithmic & Practical Approach

François Vessaz

UNIL | Université de Lausanne

FACULTÉ DES HAUTES ÉTUDES COMMERCIALES

DÉPARTEMENT DE SYSTEMES D'INFORMATION

System-level Support for Mobile Ad hoc Communications: an Algorithmic & Practical Approach

THÈSE DE DOCTORAT

présentée à la

Faculté des Hautes Etudes Commerciales
de l'Université de Lausanne

pour l'obtention du grade de
Docteur en Systèmes d'Information

par

François VESSAZ

Directeur de thèse
Prof. Benoît Garbinato

Jury

Prof. Jacques Duparc, Président
Prof. Marco Tomassini, expert interne
Prof. Patrick Eugster, expert externe
Prof. Antonio Carzaniga, expert externe
Dr. Adrian Holzer, expert externe

LAUSANNE
2013

Unil

UNIL | Université de Lausanne

HEC Lausanne
Le Décanat
Bâtiment Internef
CH-1015 Lausanne

IMPRIMATUR

Sans se prononcer sur les opinions de l'auteur, la Faculté des Hautes Etudes Commerciales de l'Université de Lausanne autorise l'impression de la thèse de Monsieur François VESSAZ, titulaire d'un Master en Systèmes d'Information de l'Université de Lausanne, en vue de l'obtention du grade de docteur en Systèmes d'Information.

La thèse est intitulée :

SYSTEM-LEVEL SUPPORT FOR MOBILE AD HOC COMMUNICATIONS: AN ALGORITHMIC & PRACTICAL APPROACH

Lausanne, le 23 mai 2013

Le doyen

[illegible]

Thomas von Ungern-Sternberg

Jury

Professor Benoît Garbinato
Professor at the Faculty of Business and Economics of the University of Lausanne.
Thesis supervisor.

Professor Jacques Duparc
Professor at the Faculty of Business and Economics of the University of Lausanne.
President of the Jury.

Professor Marco Tomassini
Professor at the Faculty of Business and Economics of the University of Lausanne.
Internal expert.

Professor Patrick Eugster
Associate Professor at the College of Science, Purdue University, West Lafayette, Indiana, USA.
External expert.

Professor Antonio Carzaniga
Associate Professor at the Faculty of Informatics of Università della Svizzera italiana, Lugano, Switzerland.
External expert.

Doctor Adrian Holzer
Research Associate at Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland.
External expert.

Université de Lausanne
Faculté des Hautes Etudes Commerciales

Doctorat en Systèmes d'Information

Par la présente, je certifie avoir examiné la thèse de doctorat de

François Vessaz

Sa thèse remplit les exigences liées à un travail de doctorat.
Toutes les révisions que les membres du jury et le-la soussigné-e ont demandées durant le colloque de thèse ont été prises en considération et reçoivent ici mon approbation.

Signature : [illegible] Date : 24.05.2013

Prof. Benoît GARBINATO
Directeur de thèse

Université de Lausanne
Faculté des Hautes Etudes Commerciales

Doctorat en Systèmes d'Information

Par la présente, je certifie avoir examiné la thèse de doctorat de

François Vessaz

Sa thèse remplit les exigences liées à un travail de doctorat.
Toutes les révisions que les membres du jury et le-la soussigné-e ont demandées durant le colloque de thèse ont été prises en considération et reçoivent ici mon approbation

Signature : [illegible] Date : 24.05.2013

Prof. Marco TOMASSINI
Membre interne du jury

Université de Lausanne
Faculté des Hautes Etudes Commerciales

Doctorat en Systèmes d'Information

Par la présente, je certifie avoir examiné la thèse de doctorat de

François Vessaz

Sa thèse remplit les exigences liées à un travail de doctorat.
Toutes les révisions que les membres du jury et le-la soussigné-e ont demandées durant le colloque de thèse ont été prises en considération et reçoivent ici mon approbation

Signature : [illegible] Date : 24.05.2013

Prof. Patrick EUGSTER
Membre externe du jury

Université de Lausanne
Faculté des Hautes Etudes Commerciales

Doctorat en Systèmes d'Information

Par la présente, je certifie avoir examiné la thèse de doctorat de

François Vessaz

Sa thèse remplit les exigences liées à un travail de doctorat.
Toutes les révisions que les membres du jury et le-la soussigné-e ont demandées durant le colloque de thèse ont été prises en considération et reçoivent ici mon approbation

Signature : [illegible] Date : 24.05.2013

Prof. Antonio CARZANIGA
Membre externe du jury

Université de Lausanne
Faculté des Hautes Etudes Commerciales

Doctorat en Systèmes d'Information

Par la présente, je certifie avoir examiné la thèse de doctorat de

François Vessaz

Sa thèse remplit les exigences liées à un travail de doctorat.
Toutes les révisions que les membres du jury et le-la soussigné-e ont demandées durant le colloque de thèse ont été prises en considération et reçoivent ici mon approbation

Signature : [illegible] Date : 24.05.2013

Dr. Adrian HOLZER
Membre externe du jury

Abstract

Version française au verso.

A mobile ad hoc network (MANET) is a decentralized and infrastructure-less network. This thesis aims to provide support at the system-level for developers of applications or protocols in such networks. To do this, we propose contributions in both the algorithmic realm and in the practical realm. In the algorithmic realm, we contribute to the field by proposing different context-aware broadcast and multicast algorithms in MANETs, namely *six-shot broadcast*, *six-shot multicast*, *PLAN-B* and a generic algorithmic approach to optimize the power consumption of existing algorithms. For each algorithm we propose, we compare it to existing algorithms that are either probabilistic or context-aware, and then we evaluate their performance based on simulations. We demonstrate that in some cases, context-aware information, such as location or signal-strength, can improve the efficiency. In the practical realm, we propose a testbed framework, namely ManetLab, to implement and to deploy MANET-specific protocols, and to evaluate their performance. This testbed framework aims to increase the accuracy of performance evaluation compared to simulations, while keeping the ease of use offered by the simulators to reproduce a performance evaluation. By evaluating the performance of different probabilistic algorithms with ManetLab, we observe that both simulations and testbeds should be used in a complementary way. In addition to the above original contributions, we also provide two surveys about system-level support for ad hoc communications in order to establish a state of the art. The first is about existing broadcast algorithms and the second is about existing middleware solutions and the way they deal with privacy and especially with location privacy.

English version on the front.

Un réseau mobile ad hoc (MANET) est un réseau avec une architecture décentralisée et sans infrastructure. Cette thèse vise à fournir un support adéquat, au niveau système, aux développeurs d'applications ou de protocoles dans de tels réseaux. Dans ce but, nous proposons des contributions à la fois dans le domaine de l'algorithmique et dans celui de la pratique. Nous contribuons au domaine algorithmique en proposant différents algorithmes de diffusion dans les MANETs, algorithmes qui sont sensibles au contexte, à savoir *six-shot broadcast*, *six-shot multicast*, *PLAN-B* ainsi qu'une approche générique permettant d'optimiser la consommation d'énergie de ces algorithmes. Pour chaque algorithme que nous proposons, nous le comparons à des algorithmes existants qui sont soit probabilistes, soit sensibles au contexte, puis nous évaluons leurs performances sur la base de simulations. Nous montrons que, dans certains cas, des informations liées au contexte, telles que la localisation ou l'intensité du signal, peuvent améliorer l'efficience de ces algorithmes. Sur le plan pratique, nous proposons une plateforme logicielle pour la création de bancs d'essai, intitulé ManetLab, permettant d'implémenter, et de déployer des protocoles spécifiques aux MANETs, de sorte à évaluer leur performance. Cet outil logiciel vise à accroître la précision des évaluations de performance comparativement à celles fournies par des simulations, tout en conservant la facilité d'utilisation offerte par les simulateurs pour reproduire une évaluation de performance. En évaluant les performances de différents algorithmes probabilistes avec ManetLab, nous observons que simulateurs et bancs d'essai doivent être utilisés de manière complémentaire. En plus de ces contributions principales, nous fournissons également deux états de l'art au sujet du support nécessaire pour les communications ad hoc. Le premier porte sur les algorithmes de diffusion existants et le second sur les solutions de type *middleware* existantes et la façon dont elles traitent de la confidentialité, en particulier celle de la localisation.

Acknowledgements

Since this thesis is composed of seven scientific articles published in conferences and journals, it is written in English both to facilitate communication of results, and the readability of this thesis due to the large number of English words in the field computer science. However, I would like to extend my thanks in French, to be more direct, spontaneous and sincere!

Comme cette thèse est composée de sept articles scientifiques publiés dans des conférences ou revues, elle est rédigée en anglais afin de faciliter non seulement la communication des résultats, mais aussi sa lisibilité en raison du nombre important de mots anglais dans le domaine de l'informatique. Cependant, je me permets d'adresser mes remerciements en français, afin d'être plus direct, spontané et sincère!

Autant le dire tout de suite, j'ai passé cinq années formidables, d'août 2008 à août 2013, à faire de la recherche sur les thèmes de l'informatique, des systèmes distribués, des mobiles et bien d'autres sujets, pendant que l'industrie et la société vivaient l'évolution vers *l'informatique mobile* depuis le 29 juin 2007... En préambule à mes remerciements, je tiens à le dire clairement au sujet de mon doctorat: *"si c'était à refaire, je le referai!"*

En premier lieu, je remercie vivement mon directeur de thèse, le Professeur Benoît Garbinato, d'avoir supervisé ce travail doctoral. En tant qu'enseignant, ses cours de bachelor ont éveillé en moi un grand intérêt en ce qui concerne la programmation en particulier et l'informatique en général. Durant mon master, au fil des projets, sa manière de travailler et ses qualités humaines m'ont convaincu de m'inscrire en thèse et de rejoindre son équipe. Son encadrement parfait, tout au long de ces cinq années de recherche, a ainsi grandement contribué à la réussite de cette thèse. Je lui adresse un merci tout particulier pour les nombreuses relectures et discussions des articles produits!

Je remercie également le Docteur Adrian Holzer, mon principal co-auteur, avec qui j'ai toujours eu beaucoup de plaisir à travailler. En tant que doctorant *senior*, puis postdoc, son aide et ses conseils se sont toujours révélés précieux. Son rythme de travail et sa motivation, m'ont aidé à avancer de manière continue tout au long de cette thèse. Je lui souhaite une belle carrière académique et un jour de devenir "Prof. Ader"!

Un grand merci aussi aux autres membres du Jury qui ont pris le temps de lire attentivement ma thèse : les Professeurs Marco Tomassini, Patrick Eugster et Antonio Carzaniga. Leurs remarques constructives ont permis d'améliorer ce manuscrit. Merci également au Professeur Jacques Duparc d'avoir accepté de présider le Jury.

Merci également aux deux personnes avec qui j'ai collaboré sur un article : mon ex-collègue, Denis Rochat, qui m'a aussi cédé une excellente place de bureau, ainsi qu'Arielle Moro, qui me succédera. Je lui souhaite beaucoup de satisfaction dans son travail doctoral, et de mener à bien les projets commencés!

Je tiens à remercier vivement le fonds national suisse de la recherche scientifique (FNS) qui a principalement financé mon doctorat via le projet 138092, ainsi que l'Université de Lausanne qui a complété ce financement.

Durant ces cinq années, j'ai eu la chance de bénéficier d'un excellent environnement de travail. Pour cela, je remercie vivement tous mes collègues et amis avec qui j'ai eu le privilège de partager très régulièrement (09h30 / 11h40 / 15h30 précises!) repas, cafés, ou bien même quelques roquettes... Parmi eux, je citerai tout particulièrement : Alexandre Métrailler, Ulysse Rosselet, Boris Fritscher, Thomas Boillat et Fabio Daolio. Je leur souhaite le meilleur pour leur fin de thèse! Merci aussi à Samuel Bendahan pour les discussions politiques et à Emmanuel Fernandes pour avoir transmis à la CRUNIL mes plaintes concernant le papet vaudois.

Merci aussi à ceux que j'ai croisé à un moment ou à un autre de ma thèse : Mouna Allani, Ian Rickebusch, Riccardo Bonazzi, Cédric Gaspoz, Steve Binggeli, Grégoire Bollmann, Christian Darabos et Enea Pestelacci.

Un merci particulier à Michel Schuepbach avec qui j'ai eu l'occasion de collaborer pour l'administration système, et qui m'a rendu de nombreux services.

Je remercie également les membres de l'ISI que j'ai eu le plaisir de côtoyer et que je n'ai pas encore eu l'occasion de citer ici: Prof. Christine Legner, Prof. Yves Pigneur, Prof. Thibault Estier, Shabnam Ataee, Behnaz Bostanipour, Kazem Haki, Alberto Antonioni, Pierre Buesser, Elisabeth Fournier, ainsi que tous ceux dont j'ai omis le nom, mais qui se reconnaitront ici!

En plus de mes activités universitaires, j'ai eu la chance de pouvoir compter sur quelques activités associatives grâce auxquelles j'ai aussi beaucoup appris. Ces activités, ont été pour moi l'occasion de me changer les idées lorsque j'en ressentais le besoin. Cela m'a toujours amusé de voir comme certaines de ces activités *broyardes* pouvaient susciter tantôt curiosité, tantôt incompréhension à Lausanne! Je ne peux hélas citer ici tous les noms de mes nombreux amis faisant partie de l'équipe de char des Bonzes, des Brandons de Payerne, de la Jeunesse de Corcelles-près-Payerne, de l'équipe du Giron 2009, de l'amicale ABC, du comité de l'Abbaye Union et Fraternité, du comité des Anciens Collégiens ou de la CAV, mais ils se reconnaitront!

Je remercie vivement ma famille pour leur soutien tout au long de mes études. Je pense à mon frère, Christian, à qui je souhaite bonne chance pour son doctorat, et à mes parents, Michèle et Gérald, qui m'ont toujours encouragé ainsi que payé toutes mes études. Un grand merci!

Finalement, je ne peux terminer ces remerciements sans une ligne pour Sandra. Maintenant que ma thèse est terminée, je me réjouis de partir découvrir l'Amérique du Sud en sa compagnie ces prochains mois!

Contents

Chapter 1
Introduction

1.1 Background & definitions

The field of parallel and distributed systems has experienced many changes since the early days of computing. This is mainly due to the fact that in a few decades these systems have become ubiquitous. In the 1960s, a parallel system referred to an operating system with concurrent processes communicating within a single computer. Around 1969, UNIX [14] was developed as a parallel and multi-user operating system, which is still the basis of many of today's devices. In the 1970s and 1980s, with the emergence of local area network such as Ethernet [12] and global networking via TCP-IP [3], distributed systems referred to groups of computers communicating through a network. With the inter-connection of existing networks, appears the Internet – the largest interconnected distributed system – in the late 1980s and 1990s. Since the emergence of the World Wide Web, the Internet continues its exponential expansion, especially on mobile devices and is moving towards the so-called *Internet of things* [2].

In the mid 1990s and 2000s, traditional wired networks were extended with wireless networks based on the IEEE 802.11 standards.[1] Wireless communications introduced a lot of new issues due to mobility and interferences (problems of hidden stations) among others. Until fairly recently, those issues were mainly solved in a centralized way on a fixed infrastructure, i.e. relying on an access point that coordinates the wireless network. Another type of wireless networks made a rapid expansion in the late 2000s with the emergence of smartphones: cellular networks used to transfer not only voice but data. Again such networks have solved the problems induced by their distributed nature with both a centralized architecture and a fixed infrastructure based on antennas. Due to the exponential growth of the Internet, IPv4 addresses rapidly became insufficient. Nevertheless, workarounds such as NAT allowed to put off the problem for a couple of years. But IPv4 also poorly fits to an increasingly mobile environment, so IPv6 is now accepted as the only

[1] IEEE 802.11 working group: `http://www.ieee802.org/11/`

long-term solution. With the advent of wireless ubiquitous and connected mobile devices, the Internet and distributed systems community is now facing the challenge of transitioning towards IPv6.

1.1.1 Mobile Ad hoc Networks (MANETs)

Solving distributed problems in wireless networks in both a decentralized and infrastructureless way has been broadly studied, mainly by the research community, since the appearance of wireless network in the mid 1990s [13]. This kind of networks are known as Mobile Ad hoc Networks (MANETs).[2] *A MANET is a wireless networks without a fixed infrastructure, which has a decentralized architecture.* However only few implementations are in production today due both to the fact that it is technically harder to manage a decentralized network and to the fact that operators have found no satisfactory economic model yet. However the lack of economic model does not mean the absence of interest to do research in this area. For instance, the Defense Advanced Research Projects Agency (DARPA) has funded MANETs research for a multi-million amount during the past five years.[3]

During the period of this thesis (2008 – 2013), due to the large number of smartphones and laptops equipped with wireless antennas and implementing IEEE 802.11 standards, and its IBSS mode (ad hoc mode), wireless networks without a fixed infrastructure have become promising like never before. In addition, many of these devices are now equipped with multiple sensors, such as GPS, and can thus act according to the context in which they operate. This opens a lot of opportunities for developers to create context-aware software that will simplify the life of mobile users.

This is the situation of distributed systems as described previously and more specifically of MANETs, at the time of this thesis. Our motivation to study MANETs, lies in the future and the promises of what (current) technologies are capable of. As millions of mobile and embedded sensing devices are deployed and become ubiquitous, we are moving forward the *Internet of things*. Such devices have to communicate directly between each other in a self configuring way. MANETs are an answer to such a challenge. In the following, we provide an overview of these promises, classified from the most achieved and commercialised ones to those still under development.

Self-configuring & Peer-to-Peer Networks. Ad hoc communication is used to interconnect nearby devices, such as laptops or phones, typically for file sharing or synchronizing. This type of communication allows users to easily share their content

[2] IETF MANET working group: `http://datatracker.ietf.org/wg/manet/`
[3] DARPA's budget: `http://www.darpa.mil/newsevents/budget.aspx`

without having to configure or connect to a network and to find other devices. Several applications are already implemented, for example, AirDrop which allows Apple's OS X users to find nearby computers for file sharing, even if they are not connected to any existing network.[4]

Vehicular Ad hoc Networks (VANETs). VANETs are useful to improve security and to disseminate information to car drivers. They are one of the areas of MANETs that has benefited the most from the interests of academic researchers. While only a few papers were published in 2004, there are around 3'000 papers on VANETs per year that have been published by the research community since 2009.[5] However, although there exist a lot of literature on VANETs, no VANETs are deployed on real roads today. Nevertheless several car manufacturers are involved in VANETs projects.[6]

Wireless Sensor Networks (WSN). Wireless sensor networks are a kind of ad hoc network used to collect – and sometimes aggregate – data about physical parameters (such as temperature, pressure, etc.) captured by sensors. Wireless sensor networks are used to monitor all kinds of information such as air quality in cities, deformation in buildings, water quality or fire detection. Examples of wireless sensor networks are liners carrying refrigerated containers whose temperature is monitored via a sensor network and then transmitted to operators. Data is propagated or routed throughout the wireless sensor network to a specific gateway node, which can then be accessed from outside the sensor network. Commercial solution using such technology already exist but are often based on the IEEE 802.15.4 personal area networks (LR-WPANs) – a standard for low data rate that aims at preserving battery life.[7] Unlike the IEEE 802.11 standard, IEEE 802.15.4 is not widely distributed with standard hardware (such as laptops or smartphones) but requires specific hardware.

Infrastructureless communications. In hazardous situations such as natural disasters or wars, MANETs can be quickly deployed to replace damaged communication infrastructure. Several systems were designed by academics for rescuers after the occurrence of a natural disasters but, to our knowledge, no implementations are currently commercialized. Even after events like the Katrina hurricane or the tsunami in Fukushima, mobile cellular base stations, which are infrastructure-based, were used rather than MANETs.[8] MANETs are currently used in operations zone such as Iraq or Afghanistan and the DARPA is funding projects to develop ad hoc

[4] AirDrop: `http://support.apple.com/kb/PH11376`
[5] Data retrieved from `http://scholar.google.com/`
[6] Car 2 Car Communication Consortium: `http://www.car-to-car.org`
[7] IEEE 802.15.4 work group: `http://www.ieee802.org/15/pub/TG4.html`
[8] Thomas Wilhelm's presentation at DEFCON 19, 2011: `https://media.defcon.org/dc-19/presentations/Wilhelm/DEFCON-19-Wilhelm-Staying-Connected.pdf`

smartphones for such use.[9] During the riots of 2010 in Egypt, Syria and Libya, the governments shut down the communication infrastructure to stop dissemination of ideas through social networks. To counter this censorship, the Obama's administration is currently leading a project named *shadow Internet* to provide Internet access to dissidents through a *mesh network* – a MANET with only a few nodes connected to the Internet where messages are routed in an ad hoc manner between nodes until a gateway node to Internet is reached.[10]

Robotic & Autonomous Systems. There already exists multiple robot prototypes communicating through MANETs [1, 11, 15]. Examples of such prototypes are search-and-rescue robots or battlefield robots communicating in an ad hoc way to coordinate. More recently, swarms of flying robots communicating and coordinating through an ad hoc network have been prototyped. Such robots have more drastic requirements on communication performance, because of their realtime constraints.[11] For example, quadrocopters robots, may be used for surveillance or for providing wifi in an area through a mesh network.

1.1.2 Communication in MANETs

In a MANET, each device connected to the MANET is called a *node*. Nodes are equipped with a radio transmitter such as IEEE 802.11 (WiFi). The node has a *transmission range* that depends on the power and frequency of its radio, and various environmental factors, such as interferences and topology. When a node transmits a message, all other nodes in its transmission range receive the message if there is no interference. This is called a broadcast at the media access control (MAC) level and all nodes receiving the message are one hop away from the sender. If a node is outside the transmission range of the sender and must receive the message, intermediate nodes have to route the message. This is known as *multi-hop* communication.

1.1.3 System-level support for MANETs

The concept of system-level support has a very broad semantics, ranging from abstract programming paradigms and concepts to concrete practical tools. There exist two main communication paradigms inside a network. Nodes can communicate using *message passing* or *distributed shared memory* (DSM). Message passing consists

[9] `http://defensesystems.com/articles/2012/01/20/darpa-saic-smart-phone-mobile-ad-hoc-network.aspx`
[10] `http://www.nytimes.com/2011/06/12/world/12internet.html`
[11] Swarms project: `http://www.swarms.org` & TED talk `http://www.ted.com/talks/vijay_kumar_robots_that_fly_and_cooperate.html`

in exchanging – synchronously or asynchronously – messages. In DSM, nodes communicate through a shared naming scheme, which can be shared memory addresses, shared objects or even *tuple space*: a shared repository of tuple. As today's hardware and standards are largely made for packet switched network, we believe that the message passing paradigm is a more suitable communication paradigm for building communication support for MANETs.

Conceptual approach. From a conceptual perspective, software support is typically provided as distributed algorithms in the message passing paradigm. Thus, routing and dissemination of messages inside a network are two key issues. *Unicast* is a one-to-one transmission abstraction, *multicast* is a one-to-many transmission abstraction, and *broadcast* is a one-to-all transmission abstraction. These transmission abstractions are primitives used to build more complex protocols or applications on top of them. Routing algorithms define *where* and *how* a node should forward the messages it receives. As MANETs are decentralized and infrastructureless by nature, all nodes have to share the responsibility to route messages. A key question is how to evaluate the performance of such algorithms? *Effectiveness* is measuring the reliability of an algorithm, i.e., the ratio of nodes effectively delivering the messages that they theoretically should deliver. *Efficiency* is measuring the message load of an algorithm, i.e., the number of messages required for a transmission. More complex algorithms are not just determinist or probabilistic. In order to improve their performance, *context-aware* algorithms may base their decisions on information from a node context. Such information can be for example the location of the node. But this introduces problems at the user-level concerning the privacy of its location data. Another kind of context-aware information is the signal power used for transmitting messages. As mobile devices have to deal with constrained resources, such as power and bandwidth, algorithms have to save such resources.

Practical approach. From a practical perspective, system-level support is typically provided to developers as software tools such as libraries or frameworks. Libraries typically implement communication protocols that will fit inside a protocol stack. Ideally, these protocols should be hosted inside the kernel space of the operating system (OS) for performance reasons and be accessible to all applications running on the device. However, the OS drastically limits access to the kernel and developing kernel-extension tends to introduce instability into the system. In this thesis, we have chosen to solve this problem by providing support to developers at a level between the applications and the system in the form of a specialized *middleware*. A middleware is by definition a software layer – often packaged as a library – deployed on all the nodes to support communication between them. Traditionally, when proven useful, such libraries are eventually moved to the OS kernel space and become standard services. Another kind of software tools are development and testing tools. Because it is especially difficult to deploy software, run it and collect data about its execution on geographically scattered devices, providing adequate tools and testbeds can help developers to go beyond just simulating their applications or algorithms.

1.2 Problem statement

The overall question of this thesis is the following: *what system-level support do developers need for mobile ad hoc communications?* This question can be split into six more precise sub-questions.

Q1: *Which are the different message dissemination protocols in MANETs?*

Many dissemination protocols have been proposed for MANETs using various techniques such as probabilities or waiting times to decide whether a message should be retransmitted or not. Other protocols use location or signal strength to decide wether to retransmit a message or not. It is difficult to compare those algorithms because often no implementation is available and they are not expressed with the same formalism. Our goal is to write those algorithms in a same way, in order to compare their structure and then contrast their performance in different deployment scenarios.

Q2: *Can context information, such as location, be used to improve message dissemination in a MANET?*

Context-awareness is the ability to collect information about the environment and acting accordingly. In recent years, mobile devices have become increasingly context-aware thanks to embedded sensors such as GPS. Thus context-aware applications and algorithms take advantage of contextual information in order to improve their performance. Message dissemination in a MANET can be compared to the dissemination of a rumor in a crowd: the initial node sends a message, which is then probabilistically propagated by other mobile devices next to it. Our intuition is that context-awareness, especially location, can help to improve the effectiveness and efficiency of a broadcast by selecting the right nodes to propagate the message.

Q3: *How to save battery while disseminating a message in a MANET?*

Some devices have the possibility to modify the power of their radio signal when they send a message. When it comes to message dissemination, our intuition is that there is a way to save battery by adequately selecting nodes retransmitting the message with a large transmission range and other nodes retransmitting it with a smaller transmission range.

Q4: *How to fine-tune message dissemination protocols in MANETs depending on nodes density?*

Our intuition is that the behaviour of a node should be different depending on the nodes density in the area where it is located. If the density is high, reliability of communication can be ensured with only a small percentage of nodes which have to retransmit messages. On the other hand if the density is sparse, many nodes might have to retransmit messages to avoid that message dissemination stops due to nodes receiving messages but deciding to discard retransmission.

Q5: *What are the differences between a real MANET implementation and MANET simulation for performance evaluation?*

When it comes to evaluate the performance of MANET protocols, until now researchers have had essentially the choice between the reproducibility offered by simulators and the accuracy offered by a testbed relying on a MANET implementation. On the positive side, simulators are widely used by the research community and their source code is generally accessible online, which makes simulation-based evaluations fairly reproducible. Unfortunately, as they rely on the modeling of complex physical and logical parameters, it is difficult to draw general conclusions about the behavior of such protocols in real and complex settings. Testbeds, on the contrary, rely on real mobile ad hoc networking and therefore tend to offer a high-level of accuracy. For this very reason however, they also tend to impose a high development and deployment barrier. For our research, we needed a testbed framework to make an evaluation of algorithms not only based on simulations. So, our idea was to implement a testbed framework based on a real MANET both to compare results with simulations results and to use collected data from the MANET implementation, to validate or complete existing models used for simulations.

Q6: *Which are the existing privacy mechanisms for context-aware systems?*

As context-aware mobile applications become increasingly common, privacy issues become a major concern for users. Users are faced with a trade-off. On one hand, releasing information about their context allows them to take advantage of new services. On the other hand, releasing such information raises privacy issues about the potential inadequate or even malicious use of this information. Our intuition is that a middleware can help to provide to users guarantees about the use of their data. Our goal is to survey existing middlewares and the mechanisms they use to enable users privacy.

1.3 Organization & Structure

This thesis is written as a collection of articles published in conference proceedings or journals in Computer Science and Information Systems. These articles are the results of my five years of research around MANETs within the Distributed Objects Programming LAB (DOPLab). The main advantage of this structure is that each chapter can be read independently of the others. The drawback is that this structure may introduce redundancies or variations in the terms or definitions. The goal of this introduction is precisely to mitigate this drawback. After a short discussion about the methodology we used (Section 1.4), we sort the articles composing this thesis in three sets of contribution. Firstly, we provide a *comparison of system-level support* for mobile ad hoc communication and privacy issues related to MANET (Section 1.5). Secondly, we provide *context-aware dissemination algorithms* in MANETs (Section 1.6). We are particularly interested in broadcast and multicast communications. We also present an open-source *testbed framework* for communication in real MANETs (Section 1.7). Finally, Chapter 9 concludes this thesis by putting into perspectives various research directions that remain open in the field of MANETs.

1.4 Methodology

Since this thesis provides both algorithmic and practical contributions, we have to distinguish the methodologies used for each types of contribution. For the algorithmic contributions, our methodology is mainly based on analysis and simulations. We measure the performance of algorithms on various topologies possibly subject to dynamic factors. Then, we analytically review the results obtained in order to understand what are the key factors influencing performance. The performance indicators that we consider are (1) the proportion of the nodes that deliver a disseminated message; (2) the number of messages necessary to achieve dissemination with a high probability; and (3) the number of communication rounds necessary to achieve dissemination with a high probability. This approach is broadly used for MANET algorithms in the literature, but suffers a lack of accuracy under certain deployment settings due to the intrinsic limitations of simulations. For practical contributions, we implemented an open-source testbed framework for MANETs. The methodology for this contribution was more along the lines of design science and experimental approaches.

1.5 Part I: Comparison of system-level support

We contribute to the state of the art of this field of research with two surveys. The first is a survey and a comparison of broadcast algorithms in MANETs [5]. The second is a survey on privacy mechanisms provided by context-aware middleware in both MANETs and infrastructure-based networks[10].

1.5.1 Chapter 2: Comparison of broadcasts algorithms in MANETs

This contribution addresses **Q1:** *Which are the different message dissemination protocols in MANETs?* and **Q2:** *Can context information, such as location, be used to improve message dissemination in a MANET?* It is based on a paper entitled "Context-aware Broadcasting Approaches in Mobile Ad-hoc Networks" [5]. Our goal is to compare different – context-aware or not – broadcasting approaches in MANETs and to evaluate their respective performance. In recent years, as more and more mobile devices have become context-aware, several broadcasting algorithms have been introduced that take advantage of contextual information in order to improve their performance. We distinguish four approaches with respect to context: (1) context-oblivious approaches, (2) network traffic-aware approaches, (3) power-aware approaches, and (4) location-aware approaches. In context-oblivious approaches, broadcasting algorithms are determinist or probabilistic. In network traffic-aware approaches, broadcasting algorithms use a *wait-and-count-retransmissions* mechanism to decide wether or not to retransmit a message. Power-aware approaches are based on the signal strength of a received message and location-aware approaches on the location of the nodes. We perform both an algorithmic comparison and a performance comparison. The algorithmic comparison expresses every algorithm in the same pseudo-code formalism to compare their behavior. Then the performance comparison presents the results in terms of effectiveness and efficiency of these algorithms in different scenarios.

1.5.2 Chapter 3: Privacy mechanisms in context-aware middlewares

This contribution answers **Q6:** *Which are the existing privacy mechanisms for context-aware systems?* It consists of a survey of privacy-enabling middleware entitled "Middleware for Location Privacy: an Overview" [10]. Building appropriate privacy sensitive middleware is a challenging endeavor, especially for context-aware applications where location enables new services on the one hand, but on the other hand those services compromise privacy. To answer **Q6**, we survey existing middleware solutions along two dimensions: their privacy mechanisms and their architecture. We categorized fourteen platforms: five rely on access control, four on blurring, four on anonymizing and two on labeling. Access control ensures that only authorised user can access another user data. Blurring consists in intentionally providing less accurate information about an user. Anonymizing consists in providing accurate information, but in a way that this information cannot be linked to a specific user. Labeling restricts the operation that a user can perform on data.

1.6 Part II: Context-aware dissemination algorithms

In our algorithmic contributions to mobile ad hoc communications, we devise context-aware algorithms for broadcast and multicast in MANETs. We describe those algorithm in pseudocode and we evaluate their behaviour with simulations.

1.6.1 Chapter 4: Broadcast in MANETs using context-aware information

Six-shot broadcast is one of our contributions to **Q2:** *Can context information, such as location, be used to improve message dissemination in a MANET?* This algorithm was first described in "Six-shot Broadcast: a context-aware algorithm for efficient message diffusion in MANETs" [4]. *Six-shot broadcast* is a context-aware message diffusion algorithm that uses location information to fine-tune broadcasting in MANETs. Its originality resides in the scheme it uses to decide whether or not to forward a given message. This scheme is based on a widespread mechanism known as *wait-and-see.* With this mechanism, when a message m is received, a node initiates a waiting time during which it looks for retransmissions of m. When the waiting time elapses, m is forwarded unless the number of observed retransmissions is exceeding a predetermined *threshold. Six-shot broadcast* assigns a different waiting time depending on the geographical location of nodes. The idea is that before a node sends a message m, it associates six geographical *targets* to it. Then among all nodes that receive m, only those located closest to a target should forward m. After describing the operations of *six-shot broadcast*, we compare its performance to the performance of other – context-aware or not – algorithms.

1.6.2 Chapter 5: Multicast in MANETs using context-aware information

Chapter 5 contains our second contribution to **Q2:** *Can context-aware information, such as location, be used to improve message dissemination in a MANET?* The multicast problem in MANETs can be solved by two naive approaches: broadcast or multi-unicast. The broadcast approach consists in broadcasting a message m containing a multicast group name to all nodes in the network. Then, only the nodes that are members of the multicast group deliver m. The other nodes merely forward m if necessary and discard it. Multi-unicast consists in sending multiple unicast messages, one to each multicast group member. Each of these approaches is optimal in terms of message overhead in an extreme case: broadcasting is optimal when all nodes in the network are part of the multicast group, whereas multi-unicast is optimal in the case where there is only one group member. In “Six-shot Multicast: A Location-aware Strategy for Efficient Message Routing in MANETs” [6], we introduce *six-shot multicast*, a location-aware *genuine multicast*[12] algorithm devised for MANETs. Six-shot Multicast is based on the same mechanism than Six-shot Broadcast using six geographical targets to select which node is the best positioned one to forward a message. But it is also based on a directed acyclic graph (DAG) to route messages only to directions where a multicast group member is located. The creation and the maintenance of this DAG implies an extra message load which can be mitigated or even neglected. The cost of the creation phase is amortized over time as it occurs only once. As the maintenance task is only local to a node, we can mitigate its cost by using mechanisms such as piggybacking, which consists in adding data in the header of another message, or using a different channel, such as a different frequency, especially for the maintenance task. Thus Six-shot Multicast aims at providing an efficient implementation for a broad range of multicast group sizes. More precisely, it aims at offering a lower message overhead than existing multicasting algorithms for similar reliability, no matter the number of multicast group members and similar efficiency to unicast and broadcast algorithms in the extreme cases where either every node is part of the multicast group or only one is part of the group. Our communication primitive can be used as a building block for popular services, such as data streaming or group communication, when the size of the actual membership is not known a priori. The particularity of *six-shot multicast* is its location-aware routing scheme, which offers improved efficiency in terms of message overhead compared to existing algorithms, for a reasonable cost in terms of reliability.

1.6.3 Chapter 6: Power-aware broadcast in MANETs

With this contribution, we specifically address **Q3:** *How to save battery while disseminating a message in a MANET?*, by presenting and evaluating a novel approach to decrease the battery consumption of epidemic information dissemination in sensor networks. This approach is presented in a paper entitled “Injecting Power-Awareness into Epidemic Information Dissemination in Sensor Networks” [7]. In essence, our strategy consists in modulating the transmission range of sensors before they send messages. Since the range modulation follows a *power-law probability distribution*, we qualify our approach to information dissemination as being *power-law*. A consequence of this strategy is that many nodes can reach few neighbors, while few nodes can reach many neighbors. To evaluate the effects of our approach, we inject the power-law range modulation into four existing epidemic algorithms and we compare

[12] A genuine multicast as defined by [8] is a multicast that is not based on a broadcast.

their performances with their original versions, based on a fixed transmission range or on a uniform distribution of transmission ranges. This evaluation shows that our *power-law* approach improves the *efficiency* of the original algorithms in terms of power consumption, with no negative impact of their *effectiveness*, measured in terms of how many nodes are reached. The efficiency – or *energy consumption* – is the total power, measured in Joules, necessary to achieve a broadcast in the whole network.

1.6.4 Chapter 7: Adaptive broadcast in MANETs

This contribution addresses **Q4:** *How to fine-tune message dissemination protocols in MANETs depending on nodes density?* Several context-aware broadcasting protocols have been proposed in order to meet this challenge, using location or proximity information in order to fine-tune retransmission decisions. However, existing protocols often target one specific setting and can reveal to be sub-optimal when settings change. Typically, optimal parameters for dense networks will differ from optimal parameters for sparse networks. "PLAN-B: Proximity-based Lightweight Adaptive Network Broadcasting" [9] aims at addressing this issue. In this paper, we propose an adaptive proximity-based broadcast protocol that offers the ability to define policies in order to adapt its parameters for different network settings at runtime based on the nodes density. Such parameters are typically thresholds defining the number of retransmissions of a same message required before discarding its retransmission. For example, if the density of nodes is high, this threshold can be increased in order to improve the efficiency with only little risk to affect the effectiveness, because more neighbours have the likelihood to achieve the dissemination of the message. Our performance evaluations show that *PLAN-B* outperforms existing static and adaptive protocols by a factor up to 2 in unknown or dynamic densities.

1.7 Part III: Testbed Framework

In the current research literature, performance evaluations of MANET-specific protocols have two problems. 1) Those performance evaluations are mainly based on simulations. Unfortunately simulations often exhibit a lack of accuracy due to the fact that models tend to over-simplify reality. 2) As for testbeds, which are real implementations and thus provide more accurate results, only a few exist and they tend to run on specific hardware. This implies that performance evaluations based on testbeds are less easily reproducible by the community because one require the same hardware to confirm the results. To address those two problems, we propose a testbed framework running on standard hardware, so MANET-specific developers can test their algorithm beyond simple simulations.

1.7.1 Chapter 8: ManetLab

ManetLab is our answer to **Q5:** *What are the differences between a real MANET implementation and MANET simulation for performance evaluation?* ManetLab is a testbed framework running on standardised hardware. It is an open-source framework for developing, configuring, running and analysing MANET-specific protocols.[13] ManetLab is described in the paper "Developing, Deploying and Evaluating Protocols with ManetLab". The goal of ManetLab is to provide both the accuracy of testbed-based evaluations and the reproducibility of simulation-based evaluations. In the context of this chapter, we also survey existing performance evaluation tools and compare ManetLab results to the results obtained with simulators. We observe that simulators provide adequate results in "simple" scenarios where there are no interferences, no mobility or no physical obstacles such as walls. But as soon as the scenario becomes more complex, simulations results tend to deviate from reality. So we advocate for the combined and iterative use of simulations and testbeds, when it comes to devise MANET-specific protocols.

References

[1] G. Antonelli, F. Arrichiello, S. Chiaverini, and R. Setola. A self-configuring manet for coverage area adaptation through kinematic control of a platoon of mobile robots. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1332 – 1337, aug. 2005.

[2] Kevin Ashton. That 'internet of things' thing. *RFiD Journal*, 22:97–114, 2009.

[3] Vinton Cerf, Yogen Dalal, and Carl Sunshine. Specification of internet transmission control program. Technical report, INWG General Note, 1974.

[4] Benoît Garbinato, Adrian Holzer, and François Vessaz. Six-shot broadcast: A context-aware algorithm for efficient message diffusion in manets. In *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I on On the Move to Meaningful Internet Systems:*, OTM '08, pages 625–638, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] Benoît Garbinato, Adrian Holzer, and François Vessaz. Context-aware broadcasting approaches in mobile ad hoc networks. *Computer Networks*, 54(7):1210 – 1228, 2010.

[13] The ManetLab source code is available on GitHub: `http://github.com/doplab/manetlab`. A binary version is also available on `http://doplab.unil.ch/manetlab`.

[6] Benoit Garbinato, Adrian Holzer, and Francois Vessaz. Six-shot multicast: A location-aware strategy for efficient message routing in manets. *Network Computing and Applications, IEEE International Symposium on*, 0:1–9, 2010.

[7] Benoit Garbinato, Denis Rochat, Marco Tomassini, and Francois Vessaz. Injecting power-awareness into epidemic information dissemination in sensor networks. *Future Generation Computer Systems*, 26(6):868 – 876, 2010.

[8] Rachid Guerraoui and André Schiper. Genuine atomic multicast. In Marios Mavronicolas and Philippas Tsigas, editors, *Distributed Algorithms*, volume 1320 of *Lecture Notes in Computer Science*, pages 141–154. Springer Berlin Heidelberg, 1997.

[9] A. Holzer, F. Vessaz, S. Pierre, and B. Garbinato. Plan-b: Proximity-based lightweight adaptive network broadcasting. In *Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on*, pages 265 –270, aug. 2011.

[10] Adrian Holzer, Benoît Garbinato, and François Vessaz. Middleware for location privacy: an overview. In *Proceedings of the 2012 ACM Research in Applied Computation Symposium*, RACS '12, pages 296–303, New York, NY, USA, 2012. ACM.

[11] Brian B. Luu, Barry J. O'Brien, David G. Baran, and Rommie L. Hardy. A soldier-robot ad hoc network. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 558 –563, march 2007.

[12] Robert M. Metcalfe and David R. Boggs. Ethernet: distributed packet switching for local computer networks. *Commun. ACM*, 19(7):395–404, July 1976.

[13] Ram Ramanathan and Jason Redi. A brief overview of ad hoc networks: challenges and directions. *IEEE communications Magazine*, 40(5):20–22, 2002.

[14] Dennis M. Ritchie and Ken Thompson. The unix time-sharing system. *Commun. ACM*, 17(7):365–375, July 1974.

[15] Zhigang Wang, MengChu Zhou, and N. Ansari. Ad-hoc robot wireless communication. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 4, pages 4045 – 4050 vol.4, oct. 2003.

Part I
Comparison of System-level Support

Chapter 2
Context-aware Broadcasting Approaches in Mobile Ad-hoc Networks

Published in Computer Networks
The International Journal of Computer and Telecommunications Networking,
Volume 54, Issue 7, 17 May 2010.
Pages 1210 - 1228
Publisher: Elsevier
ISSN: 1389-1286

Abstract The aim of this paper is to compare different context-aware broadcasting approaches in mobile ad-hoc networks *(MANETs)* and to evaluate their respective performances. Message broadcasting is one of the core challenges brought up by distributed systems and has therefore largely been studied in the context of traditional network structures, such as the Internet. With the emergence of MANETs, new broadcasting algorithms especially geared at these networks have been introduced. The goal of these broadcasting algorithms is to ensure that a maximum number of nodes deliver the broadcasted message *(reliability)*, while ensuring that the minimum number of nodes retransmit the broadcasted message *(efficiency)*, in order to save their resources, such as bandwidth or battery. In recent years, as more and more mobile devices have become context-aware, several broadcasting algorithms have been introduced that take advantage of contextual information in order to improve their performance. We distinguish four approaches with respect to context: (1) context-oblivious approaches, (2) network traffic-aware approaches, (3) power-aware approaches, and (4) location-aware approaches. This paper precisely aims at presenting these four different broadcasting approaches and at measuring the performance of algorithms built upon them.

2.1 Introduction

During the past years we have been witnessing a massive increase of mobile devices. These devices are now ubiquitous and changed the traditional architecture of distributed systems from a centralized and wired one to a heterogeneous and frequently changing one. With the rise of laptops, smart phones and the gain in popularity of wireless networks, such as WiFi and Bluetooth, new types of network architectures like mobile ad-hoc networks appear. These architectures brought up many new challenges and opportunities in distributed computing systems, in particular with regard to networking protocols. This paper focuses on a particular kind of protocols targeting mobile ad-hoc networks, namely broadcasting protocols. We classify these protocols with respect to their usage of contextual information in order to improve their performances. This paper is at the crossing between mobile ad-hoc networks, broadcasting and context-awareness.

2.1.1 Mobile ad-hoc networks

A *mobile ad-hoc network (MANET)* is a mobile network without a fixed infrastructure, i.e., it is self-configuring and nodes connect directly to each other. Because the topology of the network may change frequently, no centralize entity can be used for message routing, thus all nodes are responsible for this task. Applications based on MANETs range from autonomous intelligent sensor networks to end-user mobile applications, such as traffic jam prevention system, information dissemination in crowds, strategic data gathering in hostile environments, and peer-to-peer mobile games.

The *transmission range* of a node is the area that is covered by its radio signal. This range varies depending on the communication technology used, e.g., Bluetooth up to 10 m, WiFi up to 250 m. In MANETs, when a node sends a message, every node in its transmission range can receive the message. This mechanism is managed by the MAC layer of the nodes. The *neighborhood* of a node is composed of all nodes in its transmission range. The *density* of a MANET is the ratio between the number of nodes connected in the MANET and the size of the geographical area in which these nodes evolve.

Because of their mobility, devices in MANETs usually have limited resources such as battery power, CPU power, memory and bandwith. So protocols and applications have to be optimized for such resources limitations.

2.1.2 Broadcasting problem in MANETs

A broadcast is the dissemination of a message to all nodes of the network. Broadcast is a very common operation in MANETs. This is useful for finding a route to a node in the MANET, or to transmit a message to all nodes in a specific range. The main problem is that often the transmission range of a node is smaller than the size of the network. The solution is to have some nodes retransmit the message to disseminate it in the network. In this case, the network is said to be *multi-hop*, a hop being the link between a node to another one in its transmission range. Usually, two properties are desirable when broadcasting some message m; they are listed below.

Reliability. As many nodes in the network as possible deliver message m. This is measured by the *delivery ratio*, i.e., the number of nodes that receive the message divided by the number of nodes in the network. An algorithm is said to be reliable if it ensures that all nodes in the network deliver m;

Efficiency. As few nodes in the network as possible forward the message m to save resources. This is measured by the $forward\ ratio$, i.e., the number of nodes that re-broadcast m divided by the number of nodes in the network.

Based on these properties, we can say that the problem of broadcasting in a MANET is to maximize both reliability and efficiency.

2.1.3 Context

In the literature, several definitions of the notion of context exist. Context variables can be classied in three categories according to [19]: *where you are,who you are with and what resources are nearby.* The notion of context can also be defined as *any information about the user and the environment that can be used to enhance the user's experiences* [8], as, *any information that can be used to characterize the situation of an entity*, an entity being about any thing [2], or as, *a combination of elements of the environment that the user's computer knows about* [5]. These different definitions all aim at specifying context in a very broad sense.

Here we rely on a more restricted definition of context centered around the communication layer, which we consider to be independent from the upper application layer and we define context as *any information coming from the physical layer, such as location sensor, battery sensor, radio transmitter, etc.*

2.1.4 Context-awareness

An algorithm is *context-aware* if it has the capability to collect informations about its environment during its execution an act accordingly. In recent years, as more and more mobile devices have become context-aware, several broadcasting algorithms have been introduced taking advantage of contextual information in order to improve their performance. In this paper, we distinguish four approches with respect to context: (1) context-oblivious approaches, where no contextual information is used, (2) network traffic-aware approaches, where information about message traffic in a node's neighborhood is used, (3) power-aware approaches, where information about the intensity of the radio signal at which a message is received is used, and (4) location-aware approaches where geographical location of senders and receivers is used.

2.1.5 Yet another survey?

Many algorithms try to solve the broadcast problem in mobile ad-hoc networks and many surveys try to classify these algorithms [1, 3, 4, 7, 13, 21, 22]. In [21] only reliable broadcasting algorithms are surveyed. In [3, 4, 13] the broadcasting problems is analyzed with a focus on building overlay networks for routing protocols, such as AODV and DSR. In contrast, we focus on the broadcasting problem and only survey gossiping algorithms, i.e., that do not build an overlay. Other surveys [1, 7, 22] classify different forwarding methods, such as probability-based methods, area-based methods, self-pruning methods or neighborhood knowledge methods. These surveys differ with ours in that we classify algorithms according to the type of information they use in their forwarding decision. This paper aims at providing a useful survey of context-aware approaches for broadcasting in MANETs with detailed algorithmic comparisons and extensive performance evaluations.

2.1.6 Contributions and roadmap

This paper puts forward two main contributions: the first is an algorithmic comparison of different context-aware broadcasting approaches (Section 2.2). The second contribution, is a performance comparison of these approaches based on simulations. Performance comparison settings are described in Section 2.3, while Section 2.4 shows the results and analyses them. Finally, Section 2.5 concludes this paper with a summary of the contributions.

2.2 Algorithmic comparison

Broadcasting algorithms all have one functionality, which is to disseminate a message m in the network so that all nodes in the network deliver m. This functionality is represented by the following two primitives:

BROADCAST(m): broadcasts a message m to every node in the network.
DELIVER(m): works as a callback when a message m is received.

Since a node may not reach all nodes in the network in one hop, some nodes may have to forward m. Thus, a major task of broadcasting algorithms is to implement an adequate forwarding decision function. In this paper, we classify broadcasting algorithms, with respect to their usage of contextual information in their forwarding decision process, in four categories: (1) context-oblivious approaches, (2) network traffic-aware approaches, (3) power-aware approaches, and (4) location-aware approaches. Before going into the algorithmic comparison for each approach, we introduce their shared communication model.

2.2.1 Communication model

Our communication model is based on a typical mobile ad-hoc network, where nodes can emit messages that can be heard by all nodes in their direct neighborhood, i.e., nodes within their transmission range. With analogy to the OSI model, the broadcasting protocols that will be described hereafter are part of the *network layer* and are accessible to the application layer through the previously presented primitives (see Figure 7.1). In order to communicate with neighboring nodes, broadcast protocols is based on the MAC layer, which is in the data link layer. The MAC layer is just above the physical link between nodes. We model its communication capabilities via the two following primitives:

MAC-BROADCAST(m): broadcasts a message m to every neighboring node, i.e., every node located in the node's transmission range.
MAC-DELIVER(m): works as a callback when a message m is received.

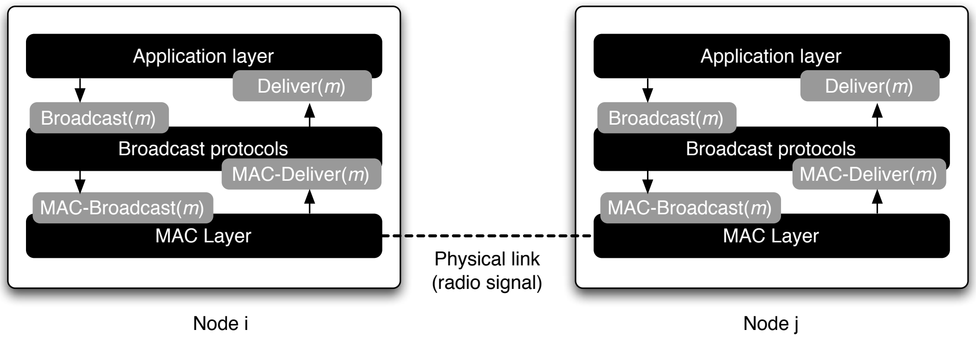


Fig. 2.1 Protocol Stack

2.2.2 Context-oblivious approaches

Context-oblivious broadcasting approaches such as *simple flooding* and *gossiping* do not use any contextual information at all in their forwarding decision process. These algorithms are very simple, as they only need a mechanism that stores received messages to determine if they were received for the first time or not.

The decision process in *simple flooding* [11] is straightforward: when a message is received for the first time, it is forwarded. This strategy is highly inefficient since the forwarding ratio is 1, but it has the advantage of being simple, quick, and reliable, as long as its inefficiency does not paralyze the network in what is called a broadcast storm [15].

The *gossip* algorithm [15], sometimes referred to as *gossip1* in the literature, is detailed in Algorithm 2.1 and works similarly to the *simple flooding* except that instead of forwarding every message, only a certain percentage of messages are forwarded (line 15). *Simple flooding* can be seen as a special case of *gossiping* where the forwarding probability is equal to 1. The choice of value p is decisive: a small value can cause the interruption of the message dissemination process and thus undermine the reliability, whereas a high value will increase the number of forwarders and might overload the network. Along that line, the efficiency of *gossiping* can be fine-tuned by varying the forwarding probability p. Unfortunately, the reliability of the algorithm declines rapidly as p decreases. A promising approach to increase efficiency and reliability is to depart from context-oblivious approaches and to consider context-aware algorithms.

```
 1: uses MAC Layer (MAC)

 2: Init:
 3:    $p \leftarrow ...$                                  {p is the probability fixed by the user}
 4:    $handledMsgs \leftarrow \emptyset$                  {list of recieved msg}

 5: To execute GOSSIP-BROADCAST($m$) :
 6:    MAC-BROADCAST($m$)                                  {broadcasts the message m}
 7:    $handledMsgs \leftarrow handledMsgs \cup \{m\}$     {adds m to the list of received messages}

 8: GOSSIP-DELIVER($m$) occurs as follows:
 9:    upon MAC-DELIVER($m$) do                            {when MAC delivers a message m}
10:       if $m \notin handledMsgs$ then                   {if m is not in the list of recieved messages}
11:          $handledMsgs \leftarrow handledMsgs \cup \{m\}$  {adds m to received messages}
12:          GOSSIP-DELIVER($m$)                           {delivers m, ¬ blocking}
13:          DECIDE($m$)

14: function DECIDE($m$)
15:    if RANDOM $< p$ then                                {if a random number is less than p}
16:       MAC-BROADCAST($m$)                               {forwards the message}
```

Algorithm 2.1 Gossiping algorithm

2.2.3 Network traffic-aware approaches

A first type of contextual information that broadcasting algorithms can take advantage of is network traffic. Unlike in context-oblivious approaches, algorithms using network traffic-aware approaches do not make their forwarding decisions on their sole internal state, but also take into account the behaviour of their neighboring nodes. The basic idea of such algorithms is to *wait and count*. When a message m is delivered for the first time, the algorithm waits for a certain amount of time and counts the number of retransmissions of the message that it hears. After the waiting time elapses, the forwarding decision is taken according to the number of retransmissions heard. The *counter-based scheme* and the *hop-count aided broadcasting* are examples of such algorithms.

The *counter-based scheme (CBS)*[1][15], detailed in Algorithm 2.2, perfectly illustrates the *wait and count* concept. Every time a message is delivered by the MAC layer, a *counter* for that message is incremented (line 10), if the message is received for the first time, a random waiting delay is set. During the waiting phase, the message counter is incremented every time a message retransmission is received. After the waiting delay, the message is forwarded if and only if the message counter is less or equal to a predetermined *threshold* (lines 16-18). This simple mechanism results in a higher number of forwarders in low node density areas and only few forwarders in high density areas. As performance results will show, this mechanism allows to dramatically increase efficiency while ensuring a high degree of reliability.

The *hop-count aided broadcasting (HCAB)* [12], detailed in Algorithm 2.3, uses a similar mechanism as *CBS* with two twists. First, instead of using message counters, *HCAB* uses boolean flags for each message, which indicate if the message must be forwarded or not (lines 19-22). Basically the flag starts off as true and can be switched to false if a message retransmission is received. The second twist is that instead of treating every message retransmission equally and switching the flag to false upon the first one received, messages each have a hop *counter* (line 6), which is incremented after each hop (line 21). When a message is received for the first time, its hop counter is stored and the flag for the message can only be switched to false if a message retransmission with a greater hop *counter* is delivered (lines 16-18). The performance of *HCAB* is not much different from *CBS* in terms of reliability and is not as good in terms of efficiency.

Note that all following context-aware algorithms build upon the *wait and count* scheme to implement their forwarding decision functions.

[1] CBS is almost identical to the algorithm in [10], also referred to as *gossip3*, except that CBS does not perform flooding for the first k hops.

```
1: uses MAC Layer (MAC)

2: Init:
3:    threshold ← ...                                  {threshold fixed by the user}
4:    handledMsgs ← ∅                                  {list of recieved messages}
5:    counter ← ⟨0, 0, ..., 0⟩     {list of message counters, indexed by message}

6: To execute CBS-BROADCAST(m) :
7:    same as lines 6-7 of Algorithm 2.1               {same as GOSSIP-BROADCAST}

8: CBS-DELIVER(m) occurs as follows:
9:    upon MAC-DELIVER(m) do                      {when MAC delivers a message m}
10:       counter[m] ← counter[m] + 1                 {increment sthe message counter}
11:       if m ∉ handledMsgs then                  {if m is received for the first time}
12:          handledMsgs ← handledMsgs ∪ {m}            {adds m to received messages}
13:          CBS-DELIVER(m)                              {delivers m, ¬ blocking}
14:          WAIT(RANDOM)                                {waits for a random delay}
15:          DECIDE(m)

16: function DECIDE(m)
17:    if counter[m] ≤ threshold then                  {if threshold is not reached}
18:       MAC-BROADCAST(m)                                {broadcasts the msg}
```

Algorithm 2.2 Counter-based scheme algorithm

2.2.4 Power-aware approaches

Power-awareness allows a node to detect the signal strength at which a message was received. As presented previously, network traffic-awareness allows algorithm to measure and select the desired message redundancy and hence dramatically reduce the number of forwarders, while keeping a reasonable reliability. The forwarders are however selected at random in the neighborhood of a sender. Intuitively, it would be more useful if nodes located at the outskirts of a sender's neighborhood would forward message instead of nodes located close to the sender, since the former can reach a greater uncovered area than the latter. Power-aware approaches can have such a feature, since the signal strength at which a message is received decreases with distance and thus can be used to compute the proximity of the sender.

The *power-aware message propagation algorithm (PAMPA)* [14] is a good example of such an algorithm. PAMPA, as detailed in Algorithm 2.4, is identical to *CBS* except for one thing: its waiting delay is not computed randomly, but according to signal strength at which a message was received (line 12). The stronger the signal, the shorter the distance from the sender, the longer the waiting delay. With this algorithm, all nodes located closest to the edge of the sender's neighborhood will forward messages.

```
1:  uses MAC Layer (MAC)

2:  Init:
3:      $handledMsgs \leftarrow \emptyset$                          {list of recieved messages}
4:      $flags \leftarrow \langle true, true, ..., true \rangle$     {list of flags, indexed by message}

5:  To execute HCAB-BROADCAST($m$) :
6:      $m.counter \leftarrow 0$                                    {sets the hop counter of the message m}
7:      MAC-BROADCAST($m$)                                          {broadcasts m}
8:      $handledMsgs \leftarrow handledMsgs \cup \{m\}$             {adds m to the list of received messages}

9:  HCAB-DELIVER($m$) occurs as follows:
10:     upon MAC-DELIVER($m$) do                                    {when MAC delivers a message m}
11:         if $m \notin handledMsgs$ then                          {if m is received for the first time}
12:             $handledMsgs \leftarrow handledMsgs \cup \{m\}$     {adds m to received messages}
13:             HCAB-DELIVER($m$)                                   {delivers m, ¬ blocking}
14:             WAIT(RANDOM)                                        {waits for a random delay}
15:             DECIDE($m$)
16:         else
17:             if $m.counter > handledMsgs[m].counter$ then        {if counter is greater}
18:                 $flags[m] \leftarrow false$                     {flag is set to false as m should not be forwarded}

19: function DECIDE($m$)
20:     if $flags[m]$ then                                          {if flag is true, m should be forwarded}
21:         $m.counter \leftarrow m.counter + 1$                    {increments the m's hop counter}
22:         MAC-BROADCAST($m$)                                      {forwards m}
```

Algorithm 2.3 Hop-count aided broadcasting algorithm

2.2.5 Location-aware approaches

Location-aware approaches have tackled the broadcasting problem in two ways. First, algorithms like *six-shot broadcast* [20] and *optimized flooding protocol* [18] use location to fine-tune the waiting time, in order to select the best possible forwarders. Second, algorithms like the *location-based scheme* algorithm [15] and the *area-based beaconless algorithm* [17] use location primarily to fine-tune the counting phase, in a similar but more sophisticated way than *HCAB*. Note that location-aware algorithms assume that every node has a positioning system available, such as a GPS.

In the *six-shot broadcast (6SB)* algorithm, message forwarding is delegated to nodes located nearby six strategic positions. These positions, called *targets* are evenly spread on the edge of the sender's neighborhood as shown in Figure 2.2 with Alice as the sender. As detailed in Algorithm 2.5, the idea is to compute a waiting delay when a message is received according to one's proximity to the closest target (lines 20-26), the closer the target the shorter the waiting delay. In order

```
 1: uses MAC Layer (MAC), Power Detection Service (PDS)

 2: Init:
 3:    same as lines 3-5 of Algorithm 2.2                    {same as Init in CBS}

 4: To execute PAMPA-BROADCAST($m$) :
 5:    same as lines 6-7 of Algorithm 2.1                    {same as GOSSIP-BROADCAST}

 6: PAMPA-DELIVER($m$) occurs as follows:
 7:    upon MAC-DELIVER($m$) do                              {when MAC delivers a message $m$}
 8:       $counter[m] \leftarrow counter[m] + 1$             {increments the message counter}
 9:       if $m \notin handledMsgs$ then                     {if $m$ is received for the first time}
10:          $handledMsgs \leftarrow handledMsgs \cup \{m\}$ {adds $m$ to the list of received messages}
11:          PAMPA-DELIVER($m$)                              {delivers $m$, $\neg$ blocking}
12:          WAIT(GETDELAY(PDS-GETPOWER($m$)))               {computes a delay and waits}
13:          DECIDE($m$)

14: function DECIDE($m$)
15:    same as lines 17-18 of Algorithm 2.2                  {same as DECIDE in CBS}
```

Algorithm 2.4 Power-aware message propagation algorithm

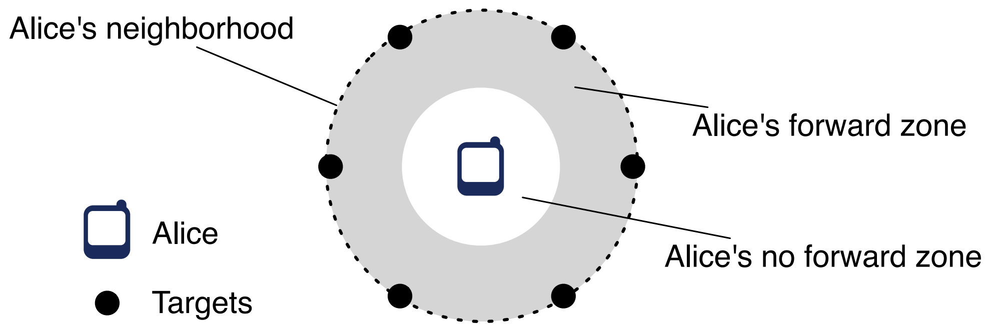


Fig. 2.2 Six-shot broadcast principles

to compute the targets (lines 27-31), the location of the sender or the forwarder is embedded in the message (line 8 and line 34). Note that only nodes located in what is called the *forward zone* can possibly forward messages, all nodes located in the *no forward zone* never forward messages (line 17). With this scheme, *6SB* exhibits improved performance results compared to all previously described algorithms.

The *optimized flooding protocol* transposes the broadcasting problem into the following geometric optimization problem: *what is the minimal number of circles to cover a certain area under the constraint that every circle must have its center on the area covered by another circle?* The answer to this problem is to divide the surface into hexagons having the size of a circle, and then to draw a circle on each

```
uses MAC layer (MAC), Location Service (LS)

Init:
    $handledMsgs \leftarrow \emptyset$                         {list of recieved messages}
    $threshold \leftarrow ...$                                 {threshold fixed by the user}
    $counter \leftarrow \langle 0, 0, ..., 0 \rangle$          {list of message counters, indexed by message}
    $nfz \leftarrow ...$                                       {size of the no forward zone fixed by the user}

To execute 6SB-BROADCAST($m$) :
    $m.center \leftarrow$ LS-GETPOSITION                       {sets m.center to the node's position}
    MAC-BROADCAST($m$)                                         {broadcasts m}
    $handledMsgs \leftarrow handledMsgs \cup \{m\}$            {adds m to the list of received messages}

6SB-DELIVER($m$) occurs as follows:
    upon MAC-DELIVER($m$) do                                   {when MAC delivers a message m}
        $counter[m] \leftarrow counter[m] + 1$                 {increments m's counter}
        if $m \notin handledMsgs$ then                         {if is received for the first time}
            $handledMsgs \leftarrow handledMsgs \cup \{m\}$    {adds m to received messages}
            6SB-DELIVER($m$)                                   {delivers m, $\neg$ blocking}
            if (LS-GETDISTANCE($m.center$, LS-GETPOSITION) $> nfz$ ) then
                WAIT(GETDELAY($m$))                            {computes a delay and waits}
                DECIDE($m$)

function GETDELAY($m$) :
    $myPosition \leftarrow$ LS-GETPOSITION                     {gets the current position of the node}
    $distance \leftarrow \perp$
    for all $p \in$ GETTARGETS($m.center$) do                  {gets the nearest target}
        if (LS-GETDISTANCE($myPosition, p$) $< distance$) $\vee$ ($distance = \perp$) then
            $distance \leftarrow$ LS-GETDISTANCE($myPosition, p$)
    return DELAY($distance$)                                   {returns the delay in function of the distance}

function GETTARGETS($p$) :
    for $0 <= i < 6$ do                                        {for all six targets}
        $p_i \leftarrow \langle p.x + sin(30 + 60 \times i) \times range, p.y + cos(30 + 60 \times i) \times range \rangle$
        $targets \leftarrow targets \cup \{p_i\}$              {adds $p_i$ to the list of targets}
    return $targets$                                           {returns the targets}

function DECIDE($m$)
    if $counter[m] \leq threshold$ then                        {if threshold is not reached}
        $m.center \leftarrow$ LS-GETPOSITION                   {sets m.center to the node's position}
        MAC-BROADCAST($m$)                                     {broadcasts m}
```

Algorithm 2.5 Six-shot broadcast algorithm

summit. Transposed back to broadcasting in MANETs, the center of each circle represents an ideal location for forwarders. Figure 2.3 depicts this idea with a central source node broadcasting a message, which will first be forwarded by three first-hop forwarders and then by six second-hop forwarders and so forth. This solution results

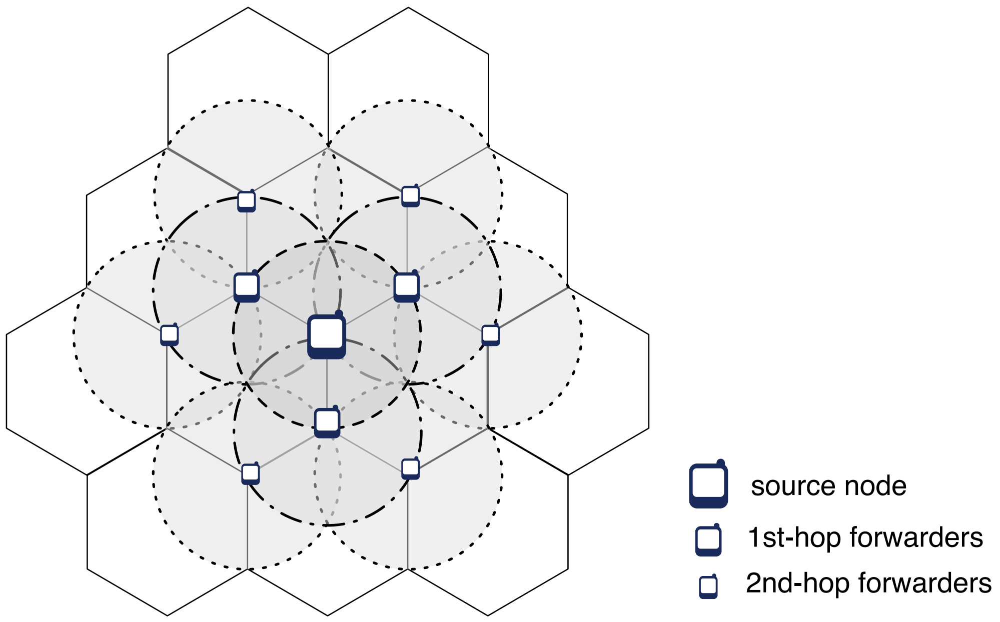


Fig. 2.3 OFP principle

in the absolute minimum number of forwarders, in order to reach all nodes within a surface in an ideal case, i.e., when all forwarders are located in ideal locations. The details of *OFP* are shown in Algorithm 2.6 and are somewhat similar to the *6SB* algorithm. *OFP* also uses a wait and count mechanism, which fine-tunes the waiting delay according to the proximity to the closest target. The difference in *OFP* is that there are three targets for the first hop (lines 17-18) and then only two targets for the following hops (lines 20-22). Targets are computed based on the location of the sender for the first hop and on the two previous senders for the following hops. The sender's location is embedded in the message in the *center* field and the previous sender in the *root* field (lines 5-6 and lines 26-27). This scheme yields the best results in terms of efficiency and very good results in terms of reliability.

The *area-based beaconless algorithm (ABBA)* [17] adopts another approach using location-awareness in that it uses location to fine-tune the counting phase. *ABBA*'s mechanism stems from the argument that if the neighborhood of a node n is fully covered by the neighborhoods of nodes who have previously sent a particular message m, then there is no need for n to forward m, since no additional node can be reached. From this argument *ABBA* imposes the following rule: a node must forward m after a

```
1: uses MAC Layer (MAC), Location Service (LS)

2: Init:
3:     same as lines 3-6 of Algorithm 2.5                          {same as Init in 6SB}

4: To execute OFP-BROADCAST($m$) :
5:     $m.root \leftarrow \perp$                                              {sets root to empty}
6:     $m.center \leftarrow$ LS-GETPOSITION                  {sets m.center to the node's position}
7:     MAC-BROADCAST($m$)                                                 {broadcasts m}
8:     $handledMsgs \leftarrow handledMsgs \cup \{m\}$      {adds m to the list of received messages}

9: OFP-DELIVER($m$) occurs as follows:
10:    same as lines 12-19 of Algorithm 2.5                           {same as 6SB-DELIVER}

11: function GETDELAY($m$) :
12:    same as lines 21-26 of Algorithm 2.5                  {same as GETDELAY in 6SB}

13: function GETTARGETS($m$) :
14:    $trgts \leftarrow \emptyset$                                    {initializes the set of targets}
15:    $c \leftarrow m.center$
16:    if $m.root = \perp$ then                         {if m is received from the initial node}
17:        for $0 < i < 3$ do                                                {creates three targets}
18:            $trgts \leftarrow trgts \cup \{\langle c.x + sin(i \times 120) \times range, c.y - cos(i \times 120) \times range\rangle\}$
19:    else
20:        $\lambda \leftarrow tan((m.root.y - c.y)/(m.root.x - c.x))$                 {creates two targets}
21:        $trgts \leftarrow trgts \cup \{\langle c.x + cos(\lambda + 120) \times range, c.y + sin(\lambda + 120) \times range\rangle\}$
22:        $trgts \leftarrow trgts \cup \{\langle c.x + cos(\lambda - 120) \times range, c.y + sin(\lambda - 120) \times range\rangle\}$
23:    return $trgts$                                                 {returns the list of targets}

24: function DECIDE($m$)
25:    if $counter[m] \leq threshold$ then                          {if threshold is not reached}
26:        $m.root \leftarrow m.center$                          {sets m.root to the previous sender}
27:        $m.center \leftarrow$ LS-GETPOSITION             {sets m.center to the node's position}
28:        MAC-BROADCAST($m$)                                             {broadcasts m}
```

Algorithm 2.6 Optimized flooding protocol algorithm

certain waiting delay, unless its entire neighborhood is covered by the neighborhood of nodes which have already forwarded the m.

Figure 2.4 illustrates this concept with two examples. In both examples, Alice initially sends m. Bob, Carol and Dave all receive m and wait for a random delay. Carol and Dave are the first ones to wake up and both decide to forward m since their neighborhoods are not fully covered. Then Bob wakes up and has to make his forwarding decision. In case ❶ his neighborhood is fully covered by Alice's, Carol's and Dave's neighborhoods so he does not forward the message, unlike in case ❷ where he must forward his message since his neighborhood is not fully covered.

Algorithm 2.7 details *ABBA*'s mechanism. Like the other location-aware algorithms,

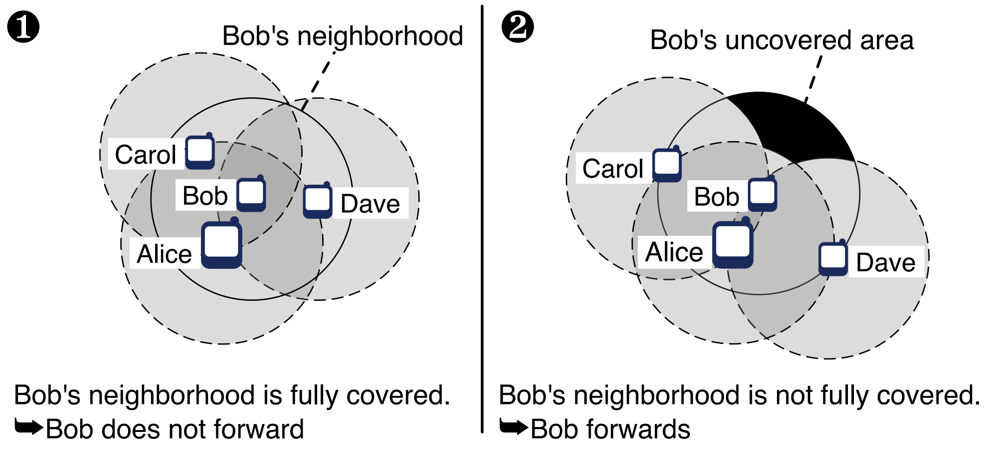


Fig. 2.4 ABBA covered perimeter

ABBA embeds the location of the sender inside the message, but unlike *6SB* and *OFP*, it keeps track of the location of initial sender of the message, as well as the sender of retransmissions of the message (line 9). To determine the waiting delay, the GETDELAY function exists in two flavors: random-based and proximity-based (line 13). The random based function is similar to the one used in *CBS*, and the proximity-based function resembles the one used in *PAMPA*, even though location is used instead of power detection. The mechanism used in *ABBA* ensures the reliability of the algorithm, at a much lower cost than *simple flooding*. The efficiency of *ABBA* is however not as high as the other algorithms studied in this paper.

2.2.6 Summary

In this section, we introduced different broadcasting algorithms, which we classified into four approaches according to their usage of contextual information in their forward decision process. Table 2.1 summarizes this classification in four approaches, namely *context-oblivious* approaches, *network traffic-aware* approaches, *power-aware* approaches, and *location-aware* approaches. One can see that all context-aware approaches build upon the network traffic-aware approach. An interesting difference of these algorithms is their forwarding patterns, depicted in Figure 2.5. Context-oblivious algorithms, such as *simple flooding* or *gossiping* have an either completely predictable (*simple flooding*) or completely random (*gossip*) forwarding pattern. Network traffic-aware algorithm such as *CBS* and *HCAB* with the introduction of

```
1: uses MAC Layer (MAC), Location Service (LS)

2: Init:
3:   $handledMsgs \leftarrow \emptyset$                                   {list of recieved msg}
4:   $positions \leftarrow \emptyset$                       {list of positions, indexed by message}

5: To execute ABBA-BROADCAST($m$) :
6:   same as lines 8-10 of Algorithm 2.5                     {same as 6SB-BROADCAST}

7: ABBA-DELIVER($m$) occurs as follows:
8:   upon MAC-DELIVER($m$) do                        {when MAC delivers a message m}
9:     $positions[m] \leftarrow positions[m] \cup \{m.center\}$        {adds the position of the sender}
10:    if $m \notin handledMsgs$ then                    {if m is recieved for the first time}
11:      $handledMsgs \leftarrow handledMsgs \cup \{m\}$              {adds m to receive messages}
12:      ABBA-DELIVER($m$)                                       {delivers m, ¬ blocking}
13:      WAIT(GETDELAY($m$))                {sets random or proximity-based delay and waits}
14:      DECIDE($m$)

15: function DECIDE($m$)
16:   if GETUNCOVEREDPERIMETER($positions[m]$) then
17:     $m.center \leftarrow$ LS-GETPOSITION                {sets m.center to the node's position}
18:     MAC-BROADCAST($m$)                                            {broadcasts m}
```

Algorithm 2.7 Area-based beaconless algorithm

Approaches / Algorithms	Context oblivious	Network traffic aware	Power aware	Location aware
Simple flooding	●			
Gossip	●			
CBS		●		
HCAB		●		
PAMPA		●	●	
6SB		●		●
OFP		●		●
ABBA		●		●

Table 2.1 Broadcasting algorithm classification

the *wait and count* mechanism, have a pseudo random forwarding pattern. The wait and count mechanism ensures that each node in the neighborhood at least receives the message a number of times equal some threshold. Power-aware algorithms, such as *PAMPA*, select forwarders based on their proximity to the sender and therefore select nodes that are closest to the neighborhood edge. Location-aware algorithms, such as *6SB* or *OFP*, go a step further and can determine ideal geographical locations of forwarders. Besides the forwarding pattern, location-awareness also allows

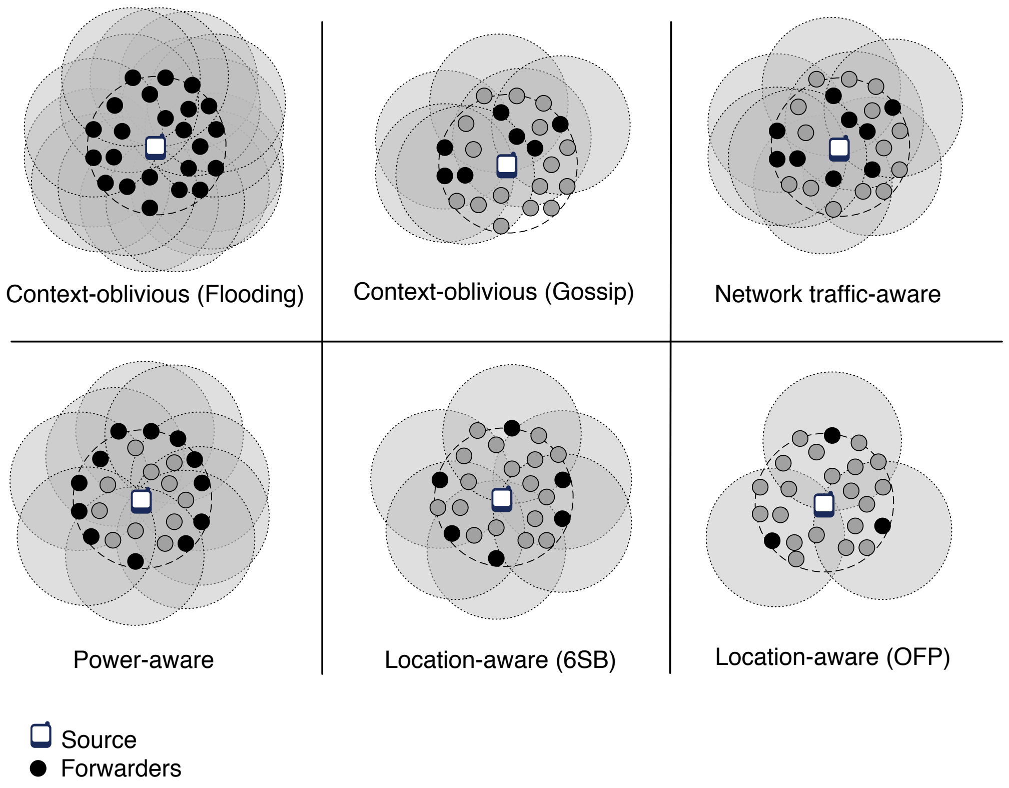


Fig. 2.5 Broadcast algorithm forwarding patterns

to devise fully reliable broadcasting algorithms, such as *ABBA*, without the need for expensive flooding.

2.3 Performance comparison settings

The second contribution of this work is to provide an extensive performance evaluation of these approaches. For our performance comparison, we used *Sinalgo* [16], a simulation framework specifically aimed at communication algorithms in wireless networks. In the following, we describe the simulation environment and parameters we used.

2.3.1 Communication scenario

The communication scenario we use for our simulations is that of a social application used to keep track of attendees of an event, such as a conference of a festival. In this scenario, every event attendee has a mobile device with IEEE 802.11 capability. All participants have an application based on similar services described in GLOVE [9]. This application is composed of three different services: (1) a polling service allowing users to vote or answer questions, (2) a radar service allowing users to locate other users at proximity, (3) a search service allowing to notify the user when someone is located nearby and matches some search criterion. We evaluate this scenario in two different settings. First, we investigate an indoor setting typical of a conference that takes place inside a building. Second, we investigate an outdoor setting typical of a festival that takes place on an open field. This outdoor setting is specifically used to evaluate the impact of increased mobility on the results of the performance evaluations.

2.3.2 Density

We define three parameters to characterize the density and the degree of connections of the network: the *transmission range*, the *map size* and the *number of nodes*. To ensure that the initial sender is able to reach all the nodes, we run a Depth First Search (DFS) algorithm before the simulation, in order to avoid simulating on networks that are partitioned. Table 2.2 summarizes the general simulations parameters we use.

Parameters	Indoor setting	Outdoor setting
Map Size	$200\ m \cdot 200\ m$	$1000\ m \cdot 1000\ m$
Map area	$4\ ha$	$100\ ha$
Transmission range	$20\ m$	$100\ m$
Min. number of nodes	160	200
Max. number of nodes	2000	600
Number of sender	1	1
Number of simulations	100	1000

Table 2.2 General simulations parameters

For the indoor setting, all nodes have a transmission range of 20 meters.[2] The map is a square of 200 meters width (4 hectares[3]) corresponding to a minimum of 10 hops in the network. To vary the node density, we vary the number of nodes in the fixed square map from 160 to 2000 (40 to 500 nodes/ha).

[2] Theoretically, IEEE 802.11 allows communications up to 140 meters outdoors and around 70 meters indoors for connections between a laptop and a router. For communication between a laptop and mobile devices, such as Apple's iPhone, our empirical tests show an indoor transmission range between 20 and 30 meters.

[3] A hectare, abbreviated *ha*, represents a 100 meter-wide square.

For the outdoor setting, the map and the transmission range are wider. The transmission range of the nodes is 100 meters and the map is a square of 1000 meters, which also results in a network size of 10 hops. In this setting we vary the node density by changing the number of nodes in the fixed square map from 200 to 600 (2 to 6 nodes/ha).

2.3.3 Mobility

In our evaluation, we use the *random waypoint* mobility model, as it is the most commonly used in the literature in the context of MANETs. An issue with this model is the fact that nodes tend to converge to the center of the map, as reported in [6]. To overcome this issue we use a torus shaped map, which avoids bound effects. In our context, we define two scenarios with mobility at walking and cycling speed. We use the walking speed in the indoor setting and cycling speed in the outdoor setting. Table 2.3 summarizes the mobility and time parameters that we use.

Parameters	Walking	Cycling
Scenario	Indoor	Outdoor
Mobility	Random Waypoint in a torus	
Delay	Uniform 0 - 10 s	Uniform 0 - 10 s
Nodes speed	Uniform 1 - 2 m/s	Uniform 5 - 7 m/s
Message transmission time	0.1 second	0.1 second

Table 2.3 Mobility parameters

We set the walking speed of our *torus random waypoint* model uniformly distributed between 1 and 2 meters per second (between 3.6 and 7.2 km/h).

For the cycling scenario, we set the cycling speed uniformly distributed between 5 and 7 meters per second (between 18 and 25.2 km/h).

In both settings, when a node reaches its waypoint, it waits for a uniformly distributed random delay, between 0 and 10 seconds. This accounts for the fact that people are not always moving. Furthermore, the message transmission time is also the same in both settings and represents the delay between a BROADCAST call on one peer and the DELIVER callback on one of its neighbors.

2.3.4 Algorithm specific parameters

Each of the previously presented algorithms has its own specific parameters that determine its behavior (see Table 2.4). We have fine-tuned these parameters to obtain the best performances for each algorithm in our scenario. The *threshold* is defined as follows: if a node receives a number of retransmissions (for a given message) that exceeds the threshold, the node will not forward that message. The *maximum delay* and the *delay function* are the mechanisms used by the nodes for the *wait and count* phase. The *no forward zone* is the ratio of the forwarder's range in which nodes do not retransmit a message.

Parameters	CBS	PAMPA	OFP
Threshold	1	1	1
No forward zone	N/A	N/A	0.4
Delay function	linear	power	linear
Max. delay	1 sec.	1 sec.	2 sec.

Table 2.4 Parameters of evaluated algorithms

2.4 Performance comparison results

We now present performance results in terms of reliability and efficiency. *Reliability* is measured by the number of nodes that received the message divided by the total number of nodes (*delivery ratio*). *Efficiency* is measured by the number of nodes that retransmit the message divided by the total number of nodes (*forward ratio*). These two ratios are presented in function of the *nodes density* (measured by the number of nodes per hectare) or in function of the *time*. For each context-aware broadcasting approach, we present the results of one representative algorithm[4]: *CBS* for *network traffic-aware*, *PAMPA* for *power-aware* and *OFP* for *location-aware*. Each simulation was run 100 times for the indoor setting and 1000 times for the outdoor setting, the following results are the mean of these executions.

We begin by presenting various performance evaluations in the indoor setting and then we present evaluations in the outdoor setting focusing on the behaviour of algorithms at different mobility level. Finally, we put all these results into perspective.

2.4.1 Indoor setting

Four different aspects of the indoor performance evaluations are presented hereafter. First, we present a glimpse of the overall results depending on the node density. Second, we zoom in on the results in low densities (40-100 nodes/ha). Unlike in high

[4] We choose these algorithms because they exhibit the best performances results compared to the other algorithms of the approach.

densities, where all surveyed algorithms are almost identical in terms of reliability, in low densities there are interesting differences and trade-offs between more reliable or more efficient algorithms. Third, we analyze the propagation time of the different algorithms in three different densities. Finally, we detail the results of the two proposed implementations of *ABBA*.

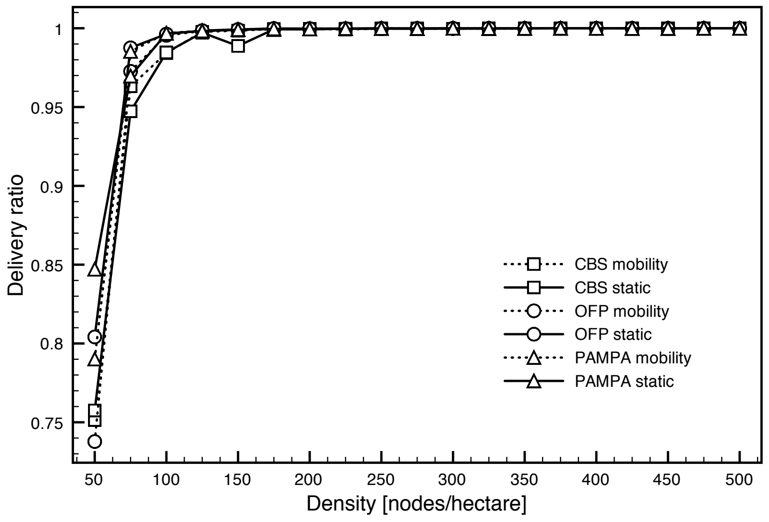


Fig. 2.6 Delivery ratios with and without mobility

Overall results. Figure 2.6 presents the *delivery ratio* of the algorithms *CBS*, *OFP* and *PAMPA* in the indoor scenario with a without and with walking mobility. All these algorithms tend to be reliable with densities above 125 nodes/ha. Indeed, there are enough nodes to propagate the message through the whole network. There are no significant differences between results with and without mobility.

Figure 2.7 presents the *forward ratio* results. Again there are no significant differences between results with and without mobility. Mobility at walking speed has no effects on the performance. This is easily explained by the speed difference between moving nodes and message propagation. The *OFP* algorithm exhibits the highest efficiency, i.e., the lowest forwarding ratio, followed by *CBS* and then *PAMPA*.

Low densities. Figure 2.8 presents the *delivery ratio* for densities under 100 nodes/ha. Such low densities exhibit more salient differences between different algorithmic approaches in terms of reliability. *CBS* is less reliable than *OFP* and *PAMPA*. *PAMPA* shows the best results for densities under 75 nodes/ha. For higher densities, *OFP* performs best.

Figure 2.9 presents the *forward ratio* in low densities. The results confirm that *OFP* has the best ratio, followed by *PAMPA* and *CBS*. It should be noted that

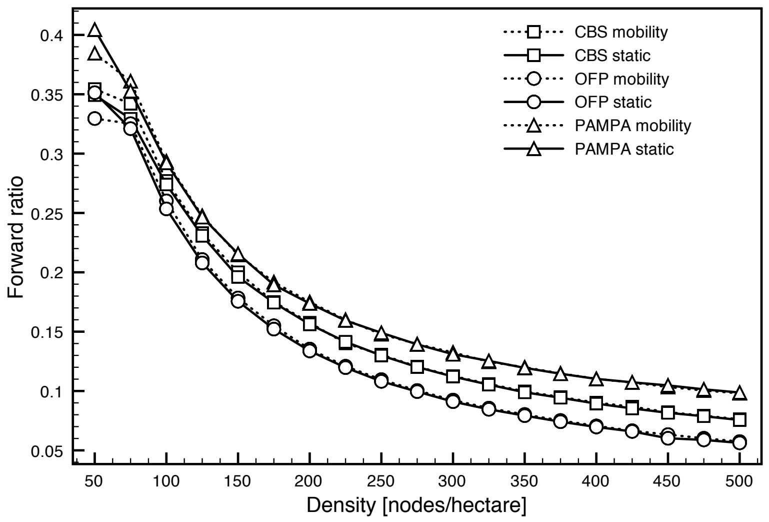


Fig. 2.7 Forward ratios with and without mobility

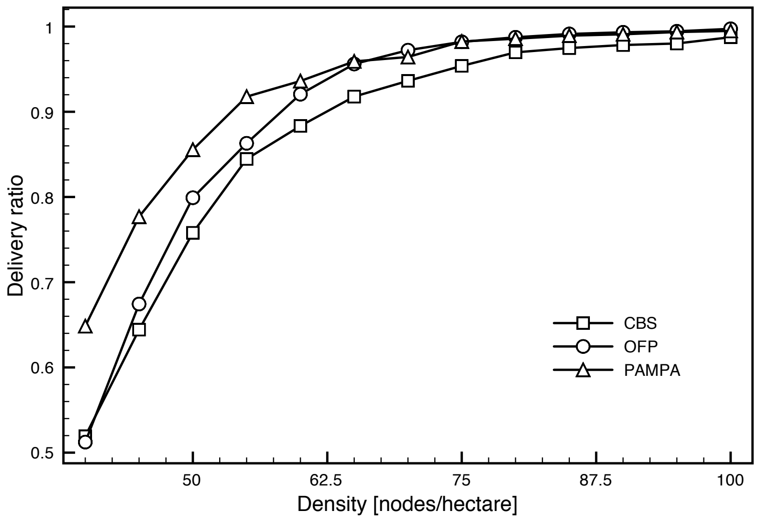


Fig. 2.8 Algorithms delivery ratios in low densities

even though *PAMPA* has a higher forwarding ratio than *OFP* it provides a better delivery ratio in very low densities.

Propagation speed. The following section analyses the propagation speed of a message in the network, i.e., the time it takes for the algorithm to terminate. In addition of our three representatives algorithms, we have add the results of the *simple flooding* algorithm. Figures 2.10-2.12 show the delivery ratios of the algorithm in densities of (respectively) 50, 100 and 500 nodes/ha.

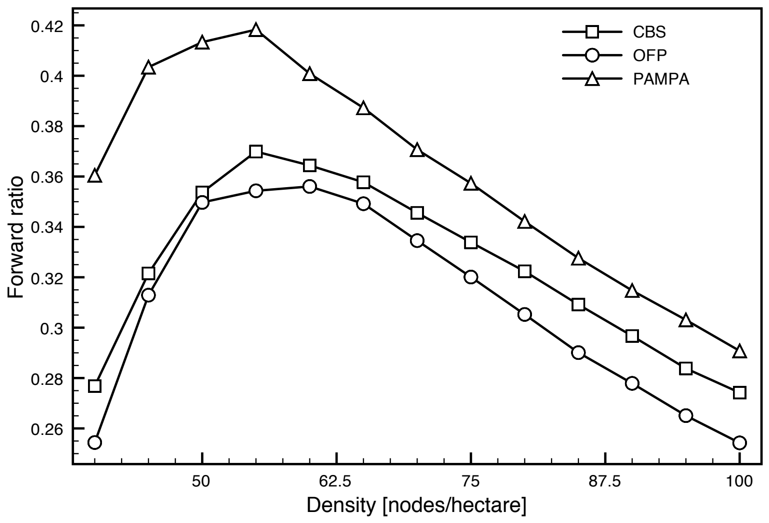


Fig. 2.9 Algorithms forward ratios in low densities

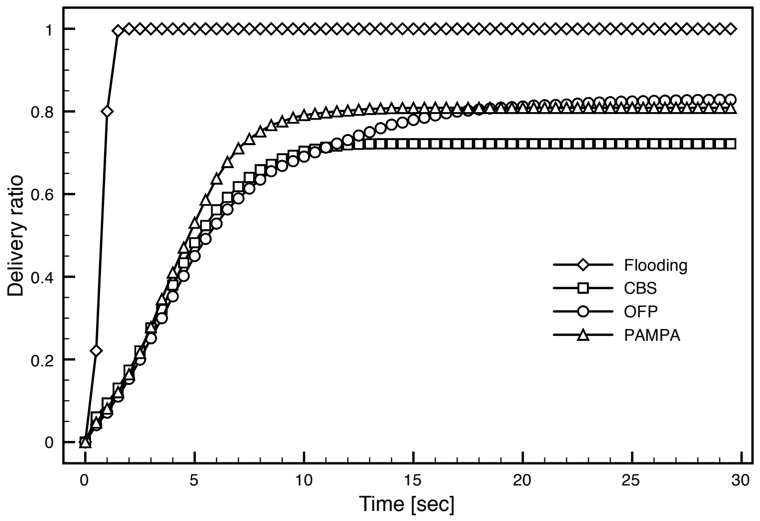


Fig. 2.10 Delivery ratios with 50 nodes/ha.

Because of its reliability, the *simple flooding* always reaches a delivery ratio of 1. This algorithm has the quickest propagation speed because there is no waiting time before the retransmission of the message. Due to the low densities of nodes in Figure 2.10, the delivery ratio of the other algorithms is between 0.7 and 0.8. In Figures 2.11 and 2.12, the node density is sufficient to allow the algorithms to reach a delivery ratio of 1. Results show that *OFP* always takes longer to terminate than *PAMPA* which in turn takes longer than *CBS*. As the densities increase, the

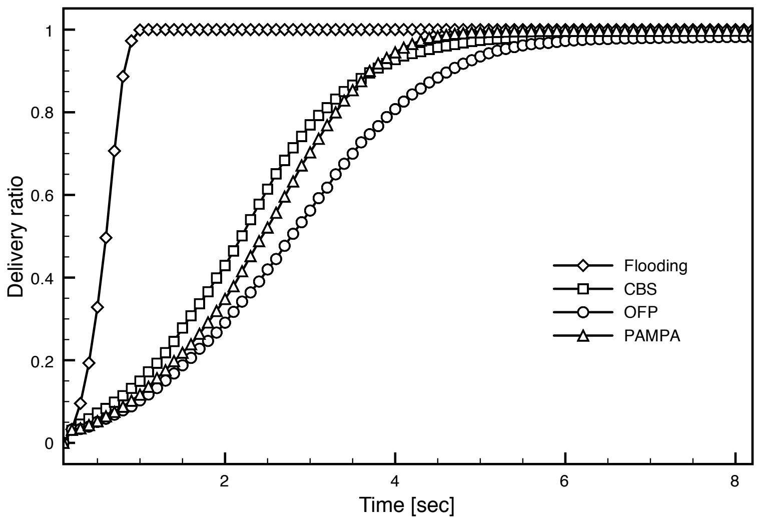


Fig. 2.11 Delivery ratios with 100 nodes/ha.

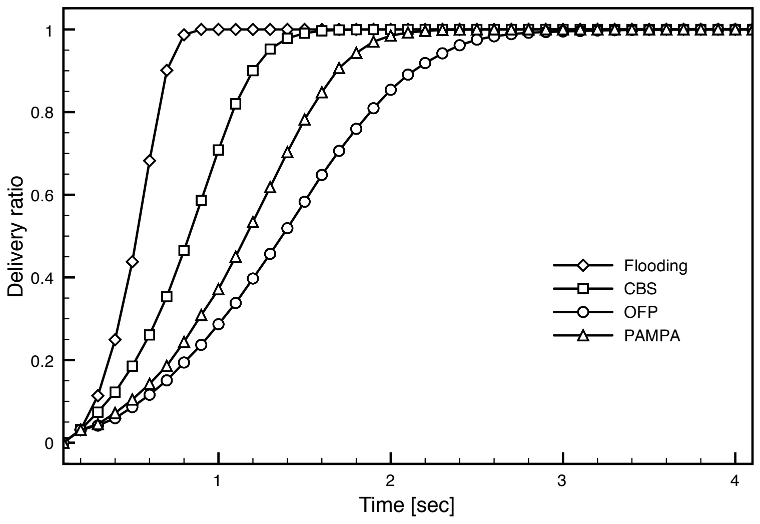


Fig. 2.12 Delivery ratios with 500 nodes/ha.

propagation time of all algorithms is reduced. These results should come as no surprise, since *OFP* assigns short waiting times only to the few nodes located nearby very specific spots in the network. In low densities, chances are that there will be no node located at the ideal spots and therefore the waiting time will be longer, hence the slower propagation speed. As *PAMPA* assigns short waiting times more generously, its waiting time is thus reduced and it enjoys a shorter propagation time.

CBS with its random waiting time attribution has the fastest message propagation speed.

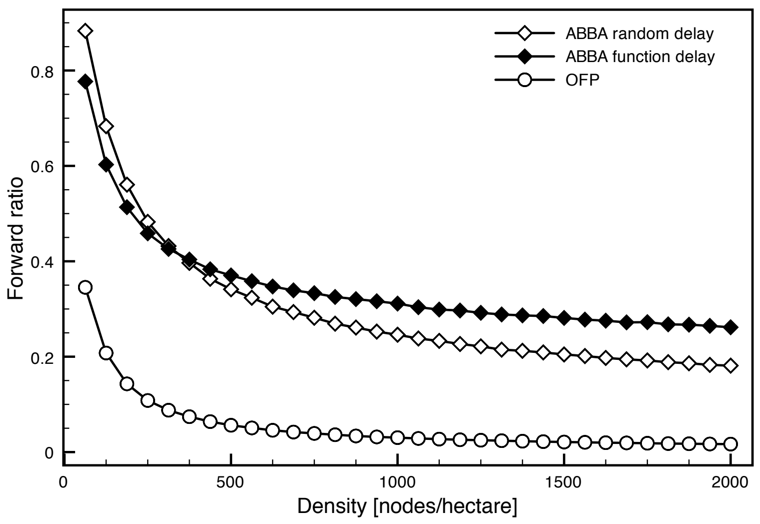


Fig. 2.13 ABBA forward ratio with and without mobility

ABBA. Figure 2.13 presents both implementations of *ABBA*, i.e., the implementation with random delay, and the implementation with proximity-based delay. The results show that the reliable delivery of *ABBA* algorithms implies a much higher forward ratio than the other location-aware algorithms, such as *OFP*. The random delay version of *ABBA* performs best with densities under 325 nodes/ha. Above 325 nodes/ha, it is the version of *ABBA* with a proximity-based delay performs best. Note that the delivery ratio of *ABBA* is always strictly equal to 1 by construction.

2.4.2 Outdoor setting results

Previously, we have seen that low mobility induced by walking speed senarios did not affect the performance of algorithms in indoor settings. In the outdoor setting, we investigate mobility at higher speeds. In order to provide a realistic senario, we introduced a cycling scenario with peers moving at speeds between 5 m/s and 7 m/s, i.e., 18-25 km/h. In addition to the cycling scenario, we add performance evaluation for speeds up to 37 m/s (130 km/h), in order to compare the effects of higher speed on the algorithms. It should be noted that such high speeds are not perfectly realistic in our outdoor scenario with a random waypoint mobility model. Nevertheless, they provide insights on how the algorithms behave in more extreme conditions.

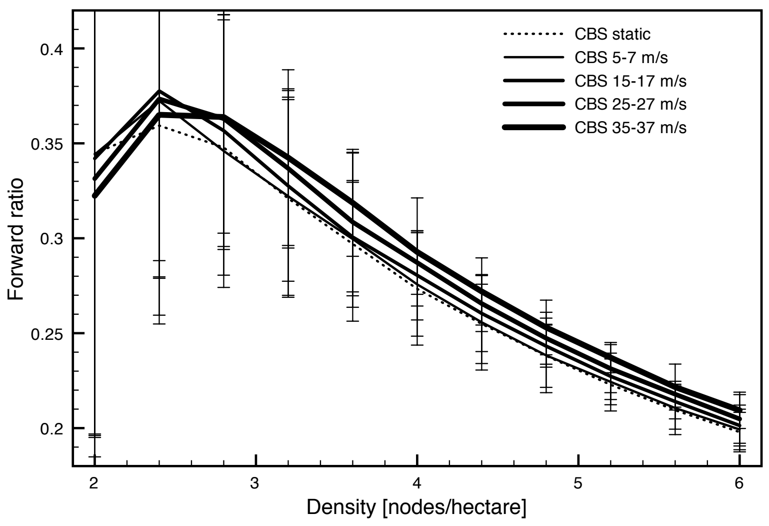


Fig. 2.14 Forward ratios of CBS

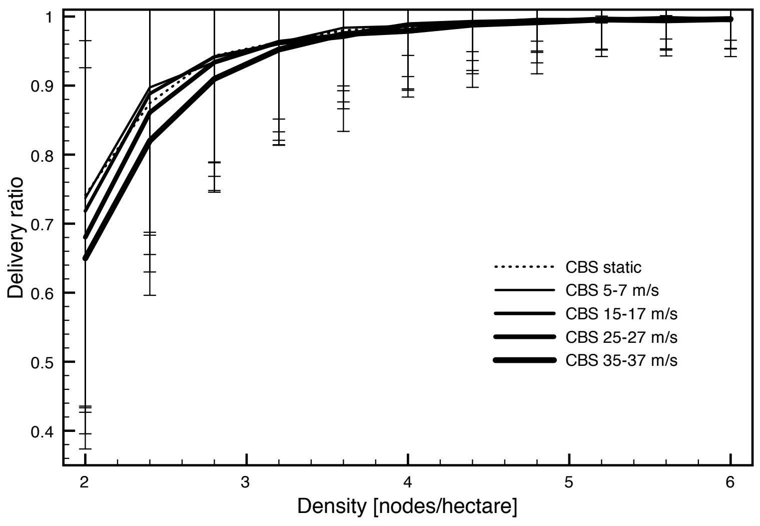


Fig. 2.15 Delivery ratios of CBS

Figure 2.14 shows the forward ratio of the *CBS* algorithm with error bars. Except in very low densities, mobility causes a slight increase of the forwarding ratio. In very low densities, mobility slightly decreases the forwarding ratio. This is due to the reduced delivery ratio. The forward ratios of *OFP* and *PAMPA* are not depicted, but behave similarly to *CBS*.

Figures2.15-2.17 shows the delivery ratios of *CBS*, *OFP* and *PAMPA*. Mobility has an effect on the delivery ratio only in low densities (less than 4 nodes/ha.). Mobility

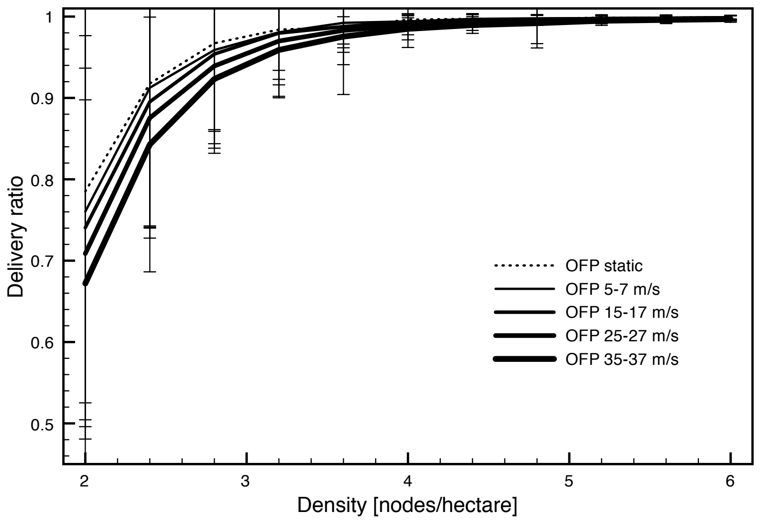


Fig. 2.16 Delivery ratios of OFP

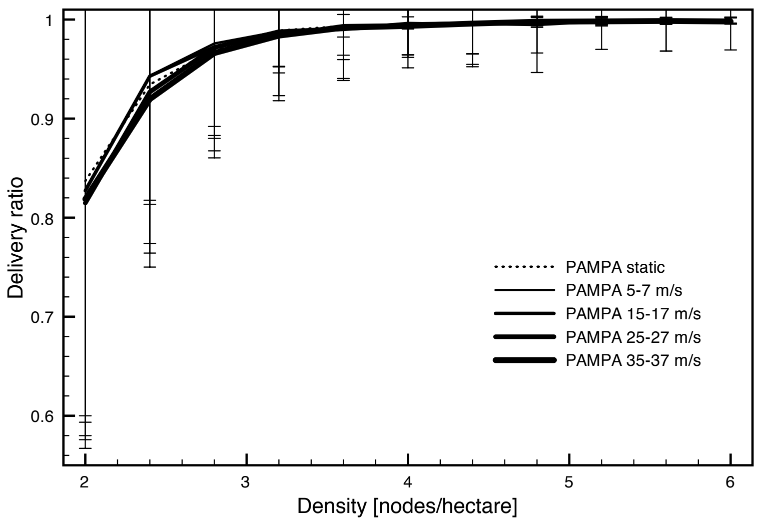


Fig. 2.17 Delivery ratios of PAMPA

causes a diminution of the delivery ratio of 0.1 for the lowest density (2 nodes/ha.) for *CBS* and slightly more for *OFP*. *PAMPA* is the most resilient algorithm and does not suffer from increased mobility. This result can be explained by the fact that *PAMPA* is the algorithm that generates the highest forward ratio compared to *CBS* and *OFP*. Increased redundancy in this case leads to increased resilience.

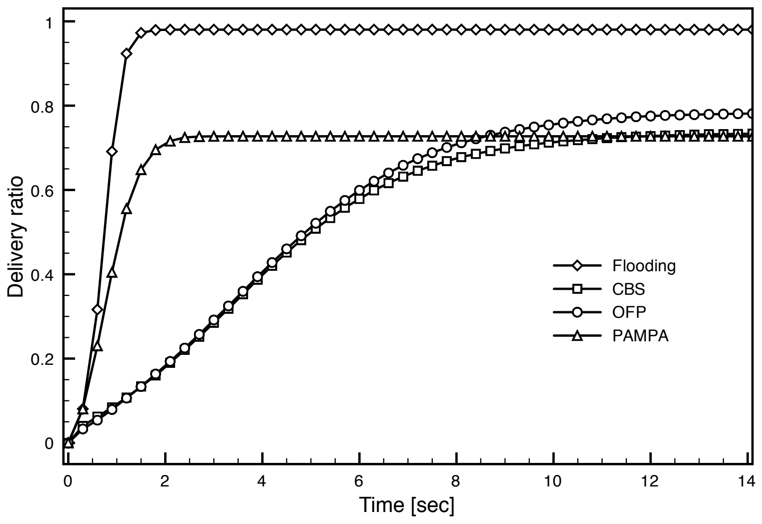


Fig. 2.18 Delivery ratios with 2 nodes/ha.

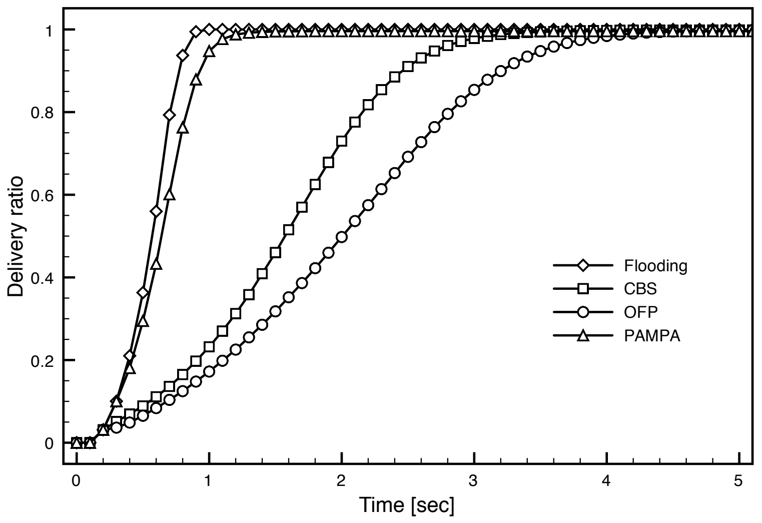


Fig. 2.19 Delivery ratios with 6 nodes/ha.

Propagation speed. Figures 2.18 and 2.19 show the propagation speed results in the outdoor scenario. In addition to our three representatives algorithms, we add simple flooding as a reference algorithm. The quickest algorithm is PAMPA, followed by OFP and CBS. We can see that the higher the density, the faster the propagation speed. In figure 2.18, only simple flooding can reach a delivery ratio of 1 due to the small number of edges in such low densities.

2.4.3 Discussion

Overall, the *network traffic-aware approach* is a very interesting one. It is implemented by algorithms such as *CBS* and *HCAB*. These algorithms are also quite simple and have performances that are comparable to those of much more complicated algorithms. The only disadvantage in comparison to *context-oblivious* algorithms is the message propagation speed in the network, which is slower due to the waiting time. However their propagation speed is higher than the propagation speed of *power-aware* and *location-aware* approaches.

Power-aware algorithms like *PAMPA* have to bear the penalty of measuring the signal strength for each received message. This penalty impacts both the algorithmic complexity and the technological complexity.[5] This approach is slightly less efficient, but more reliable, particularly in low densities or in networks with high interference. Mobility has a smaller impact on this approach in low densities with an high speed, and this approach has a good propagation time in low densities.

Location-aware have to bear the penalty of a location service (such as GPS). By selecting the best nodes to retransmit the message, *location-aware approaches*, such as *OFP* and *ABBA*, perform best: *OFP* has the best forwarding ratio (most efficient) for one of the best delivery ratios, while *ABBA*, which is reliable, has good efficiency for a delivery ratio strictly equal to 1.

2.5 Conclusion

Finding the right algorithm for broadcasting in a MANET is not a trivial task. The problem of broadcasting in MANETs is a tradeoff between reliability and efficiency. Reliability is measured by the number of nodes that deliver the message and efficiency by the number of nodes that forward the message. A maximum of nodes have to deliver the message, while we try to minimize the number of nodes that forward this message. By optimizing the number of forwarders, we save resources, such as battery power, but we increase the risk that some nodes do not receive the message.

The first contribution of this paper has to propose a classification and an algorithmic comparison of four broadcasting approaches: *context-oblivious* approaches, which make their forwarding decision based solely on their internal state, *network traffic-aware* approaches, which use contextual information about network traffic to make their decision, *power-aware* approaches, which use information about the signal strength at which messages are received, and *location-aware* approaches, which use geographical locations of nodes.

The second contribution of this paper is a performance analysis of some algorithms following each of these approaches. *Context-oblivious* approaches are very simple and

[5] This technological complexity typically translates into additional specialized hardware.

not very efficient, but provide the advantage of a quick message propagation speed in the network. *Network traffic-aware* approaches are also very simple but smarter. By monitoring the network, nodes determine if the number of messages retransmission is sufficient to ensure reliability. These algorithms exhibit a higher degree of efficiency. *Power-aware* approaches are less efficient, but exhibit the best reliability among the non-reliable algorithms, especially in low densities. *Location-aware* approaches are theoretically the best ones. Algorithms like *ABBA* exhibit the highest efficiency among reliable algorithms and algorithms like *OFP* exhibit the highest efficiency for a reasonable reliability. The disadvantages of these algorithms is in the penalty of the localization service and in the risk of failure if localization is unavailable or inaccurate. However, the growing ubiquity of GPS capabilities in mobile devices makes these approaches particularly interesting for the future.

References

[1] Performance analysis of broadcast protocols in ad hoc networks based on self-pruning. *IEEE Trans. Parallel Distrib. Syst.*, 15(11):1027–1040, 2004. Fei Dai and Jie Wu.

[2] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC'99)*, pages Pages 304–307, London, UK, 1999. Springer-Verlag.

[3] Kemal Akkaya and Mohamed Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

[4] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, New York, NY, USA, 1998. ACM.

[5] P. J. Brown. The stick-e document: a framework for creating context-aware applications. In *Proceedings of EP'96, Palo Alto*, volume Vol. 8, pages Pages 259–272. Electronic Publisher, January 1996.

[6] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[7] Thomas Heide Clausen, Laurent Viennot, Tue Olesen, and Nicolai Larsen. Investigating data broadcast performance in mobile ad-hoc networks. In *In Proceeding of Wireless Personal Multimedia Communications. MindPass Cen-*

ter for Distributed Systems, Aalborg University and project Hipercom, INRIA Rocquencourt, Fifth International Symposium on Wireless Personal Multimedia Communications, 2002.

[8] Anind K. Dey, Gregory D. Abowd, and Andrew Wood. Cyberdesk: A framework for providing self-integrating context-aware services. In *Proceedings of the Symposium on User Interface Software and Technology*, pages 75–76. ACM, 1997.

[9] Patrick Th. Eugster, Benoît Garbinato, and Adrian Holzer. Design and implementation of the pervaho middleware for mobile context-aware applications. In *Proc. of MCETECH'08*, 2008.

[10] Z.J. Haas, J.Y. Halpern, and Li Li. Gossip-based ad hoc routing. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3:1707–1716 vol.3, 2002.

[11] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM.

[12] Qing Huang, Yong Bai, and Lan Chen. Efficient lightweight broadcasting protocols for multi-hop ad hoc networks. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium*, pages 1 – 5, 2006.

[13] Martin Mauve and Hannes Hartenstein. A survey on position-based routing in mobile ad hoc networks. *IEEE Network*, 15:30–39, 2001.

[14] Hugo Miranda, Simone Leggio, Luis Rodrigues, and Kimmo Raatikainen. Removing probabilities to improve efficiency in broadcast algorithms. In *Proceedings of the 5th MiNEMA Workshop, Middleware for Network Eccentric and Mobile Applications*, pages 20–24, 2007.

[15] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping 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, New York, NY, USA, 1999. ACM.

[16] Distributed Computing Group of ETH Zurich. Sinalgo: a simulation framework for manets. *http://dcg.ethz.ch/projects/sinalgo/*.

[17] Francisco Javier Ovalle-Martnez, Amiya Nayak, Ivan Stojmenovic, Jean Carle, and David Simplot-Ryl. Area-based beaconless reliable broadcasting in sensor networks. *IJSNet*, 1(1/2):20–33, 2006.

[18] Vamsi Paruchuri, Arjan Durresi, and Raj Jain. Optimized flooding protocol for ad hoc networks. *CoRR*, cs.NI/0311013, 2003. informal publication.

[19] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, pages Pages 89–101, Santa Cruz, CA, US, 1994. IEEE Computer Society.

[20] Benoît Garbinato, Adrian Holzer, and François Vessaz. Six-shot broadcast: a context-aware algorithm for efficient message diffusion in manets. In *Proc. of DOA'08 (to appear)*, 2008.

[21] Einar Vollset and Paul Ezhilchelvan. A survey of reliable broadcast protocols for mobile ad-hoc networks. Technical report, 2003.

[22] Brad Williams and Tracy 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, New York, NY, USA, 2002. ACM.

Chapter 3
Middleware for Location Privacy: an Overview

Published in the proceedings of the 2012 ACM Research in Applied Computation Symposium (RACS'12)
San Antonio, TX, USA, October 23-26, 2012
Publisher: Association for Computing Machinery
ISBN: 978-1-4503-1492-3

Abstract As context-aware mobile applications become increasingly common, privacy issues hit headlines again. Users are faced with a trade-off. On one hand, releasing information about their location allows them to take advantage of new location-based services. On the other hand, releasing such information raises privacy issues about the potential inadequate or even malicious use of location information. Location-based tourist information is a typical service that needs to overcome this tension in order to succeed. Thus, programming such services is challenging in two aspects. First, there is the inherent burden of programming distributed context-aware software. Second, there is the challenge of programming such software in a way that guarantees user privacy. Providing adequate middleware can alleviate the programming burden. In this paper, we provide an analysis of various existing privacy-enabling middleware that can be used to provide programming support for it.

3.1 Introduction

With computing devices becoming more mobile and pervasive, a stronger interaction between an application and its changing environment opens new horizons in terms of application functionalities. Mainstream GPS-based navigation systems are good examples of how information provided to an application on its surroundings offers new kinds of possibilities. Location is one of many environmental variables that might influence the behavior of an application. The notion of *context* encompasses these variables in the broad sense.

3.1.1 Context

In the literature, context variables have been defined in different ways. Authors of [42] categorize context variables along three dimensions: *where you are, who you are with and what resources are nearby*. Similarly, the authors of [19] suggest that *any information about the user and the environment that can be used to enhance the user's experiences* is part of the *user's context*. Later, in [2], the same authors propose a definition of context as *any information that can be used to characterize the situation of an entity*, an entity being about anything. In [9], the notion of context is generalized as *a combination of elements of the environment that the user's computer knows about*. These different definitions all aim at specifying context in the very large sense. In this paper we rely on a particular kind of context typical of mobile networks, namely *location*. In a network with moving mobile devices, location is an inherently dynamic context that can be employed to enhance the user experience of an application. In that sense, it is a perfect example of privacy-sensitive contextual information.

3.1.2 Context-aware Applications & Middleware

In traditional context-agnostic applications, the contextual information introduced above is not accessible to the application code. In contrast, a context-aware application is an application that can explicitly learn about a part of its context, and act accordingly. Several authors [5, 11] defined context-awareness and surveyed context-aware systems and/or applications. From a business perspective, reports about the location-aware application market show the increase of these applications in every online application stores [48].

There exist many different location-based applications. Among them we can distinguish four main types. The (1) first type are smart information applications, which represent the focus in this paper, where relevant information or advertisements are displayed according to the user's location. Typical applications of this type inform users about surrounding Points Of Interests (POI) [1, 22], such as museums or restaurants. Smart phone applications like *AroundMe* for the iPhone propose such services too. Google recently acquired AdMob[1] to extend their Google Ad services to mobile devices. Such advertisement can be location-based if the system allows it.

The (2) second type are proximity meeting or dating applications,where users can be notified when someone matching a desired profile is located nearby. *Shockfish*[2] proposes dedicated devices with proximity meeting software for conferences. Other examples include applications like *WhosHere*, *WhosAround* or *MeetMe* for iPhone.

[1] http://www.admob.com
[2] http://www.shockfish.com

The (3) third type are location-based games which can range from simple games using location to much more complicated augmented reality game [31, 41]. Examples of games developed in an academic context are described in [7, 8, 21, 46, 47]. In the business context, location games such as *Geocaching*, *Pursuit* or *CellGuided* exist on online portals, such as Google's Android Market or Apple's AppStore. Finally, the (4) fourth type are social networking application such as *FaceBook*, *Twitter* and *Loopt* which provide location-based information too. They are a mix between all of the three previous categories.

Location-based applications have been expected to become very popular for several years [32] and with the recent advent of GPS-enabled smartphones and the explosion of the mobile application market, they are finally becoming a reality for mainstream users. Various middleware solutions have been proposed to address the issues raised by the programming of context-aware applications, some of them surveyed in [5, 27, 20] and few of them focus on privacy issues. Furthermore, security and privacy are often wrongly regarded as the same thing. This paper aims at clarifying the distinction between privacy and security in the context of location information and provides an overview and classification of existing privacy-sensitive middleware.

This paper is structured as follows. Section 3.2 defines the location privacy problem in pervasive computing and differentiates it from the security problem. Then Section 3.3 and Section 3.4 present and discuss existing middleware support to ensure privacy. Finally, Section 3.6 concludes the paper by providing insights on future research opportunities.

3.2 Defining Location Privacy

The universal declaration of human rights considers privacy to be a fundamental right and states that *"no one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks"* [38]. Many countries have incorporated these rights in their Constitution and apply these concepts of privacy to our actual society of digital information. For example, Article 13 of the swiss Constitution[3] ensures that *everyone has the right to privacy in their private and family life and in their home, and in relation to their mail and telecommunications. Everyone has the right to be protected against the misuse of their personal data.* In the context of computing, we speak of the privacy of personal data which has been an issue since transactional systems where introduced [10]. For ubiquitous computing, these questions have been broadly studied for over a decade [43, 44]. Authors of [17] define privacy as *the ability of an individual to control the terms under which their personal information is acquired and used.*

[3] Swiss Constitution: http://www.admin.ch/ch/e/rs/c101.html

One of the major concerns about privacy in ubiquitous computing is the *location privacy*. With location privacy, privacy threats can be described as *an incident in which an adversary can identify a one-to-one mapping between an individual and location information* [34]. Along this line we define *location privacy* as *the guarantee that information about the location of an identifiable user cannot be gathered by an untrustworthy party*. This guarantee does not include protection against social engineering [33].

3.2.1 Security vs Privacy

The relation between the notions of security and privacy is not obvious and deserves careful attention. Security is sometimes described as a necessary but not sufficient condition for privacy [3]. We want to better understand under which condition security issues and privacy issues arise. We believe that some settings are not subject to neither privacy nor security issues, whereas some settings are subject only to privacy issues and others are subject to both privacy and security issues.

In Figure 3.1 we illustrate these different settings. In Figure 3.1.1, a location gathering system is described. The location of user A is obtained via some location sensor typically embedded in a smart phone, then stored in a database. In such a setting where no one can access the database, there is no privacy nor security issue. In Figure 3.1.2, an external user B has access to the location database, but in this system, there is security issue if B is potentially not trustworthy and if *loc* is not public information. However, there is not de facto a privacy issue since the identity of the user is not yet associated with the location information and therefore there is no link between locations and users. It should be noted, however, that the identity of a user can in many cases be reconstructed based on user location. In Figure 3.1.3, user identities are also accessible to B, therefore there is a privacy threat if B is potentially not trustworthy and if A considers her location information private. Note that security and privacy are not absolute concepts but depend on the security and privacy policies of user A's data. Such *policies* discriminate between different types of users when it comes to accessing data.

3.2.2 Privacy Policies

Since privacy is not an absolute notion – some users might not care about location-privacy [29] – privacy-aware systems need to capture user preferences through what is called a privacy policy. Such policies are defined in the literature in the following way. *Privacy policies determine who is allowed to collect data, what data is allowed to be collected and for what purpose this data is collected* [30]. A privacy policy can also

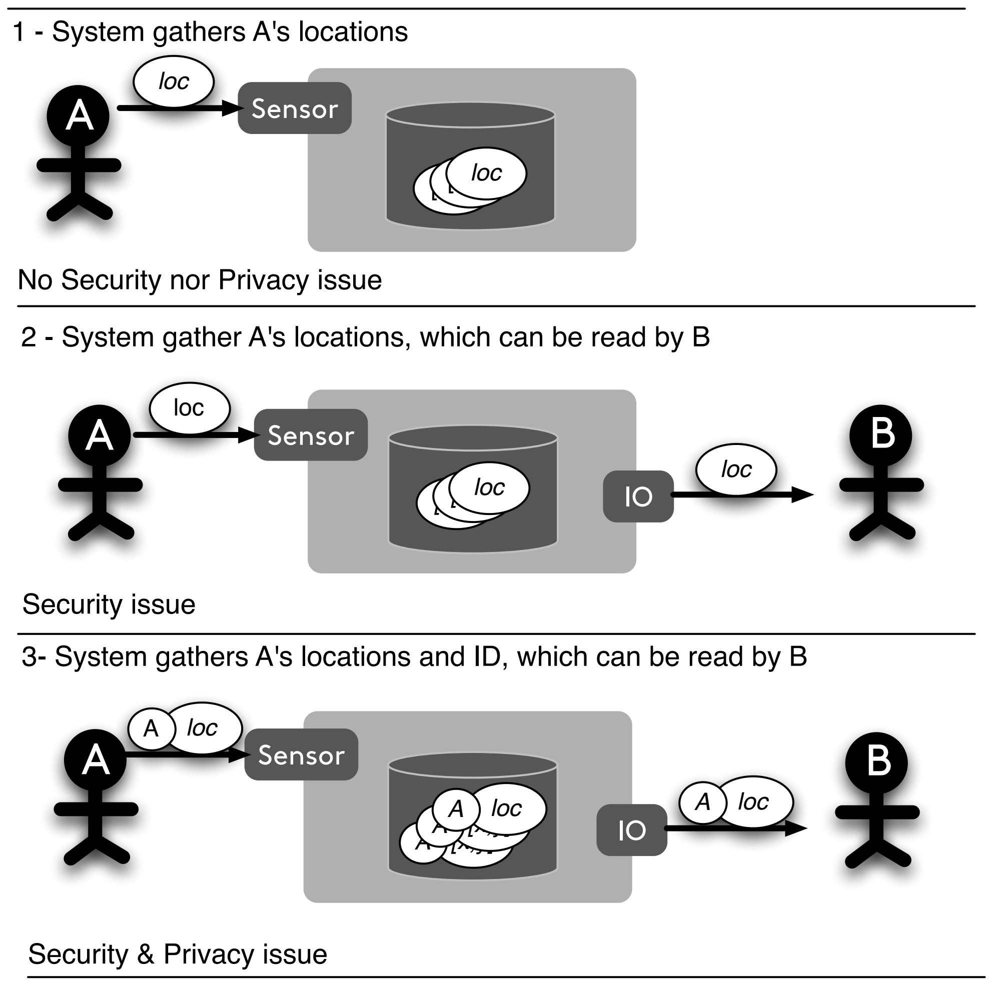

1 - System gathers A's locations
A
loc
Sensor
loc
No Security nor Privacy issue
2 - System gather A's locations, which can be read by B
A
loc
Sensor
loc
IO
loc
B
Security issue
3- System gathers A's locations and ID, which can be read by B
A
A loc
Sensor
A loc
IO
A loc
B
Security & Privacy issue

Fig. 3.1 Security / Privacy

be defined with a *permission, a subject, an object, a purpose and an obligation* [51]. In [36] authors propose several ways in which users can restrict access to their information to certain organizations or services, specifying at what time and in what context they can be accessed, as well as the location from where they can be accessed and what kind of request is allowed to access them.

Along these lines, we define a privacy policy according to four dimensions: *what* kind data is accessed, *who* accesses the data, for what *purpose* the data is accessed, and *when* is the data accessed. Since we focus on location privacy, location coupled with identity is the root of the privacy issue as described in Section 1.3. In our scenario, *what* is defined as the tuples of location and identity $D = \langle loc,\ id \rangle$. *Who* is defined as a group of applications and people $G = \{b_1, b_2, ..., b_n\}$. Typically, users can make a list of authorized people, or applications who will be allowed to perform a certain operation on the defined data in a certain context. This list can either be determined explicitly by naming authorized people, or can be determined implicitly, for example by stating that all people within a geographic location are included in the group. The *purpose* is defined as a set of operations $O = \{o_1, o_2, ..., o_n\}$. These operations represent the purpose for the access. *When* is defined as a set of contexts $C = \{c_1, c_2, ..., c_n\}$. The context here is broad. It can define for example the time at which the data can be accessed, the location from which it can be accessed, the number of peers that must be in the vicinity of the person initiating the request for the request to hold, etc. In a nutshell, a privacy policy is defined as a tuple $\rho = \langle D, G, O, C \rangle$ and represents a set of people or applications G allowed to perform a set of operations O on the set of data D if the context C is valid.

3.3 Privacy Support

When it comes to provide support for privacy, platforms use different mechanisms. In Section 3.2, we defined a privacy policy as $\rho = \langle D, G, O, C \rangle$. Here, we classify middleware solutions according to which element of the privacy policy which they aim at supporting. We take a look at the three central elements, namely data (D), group of people (G) and operations (O). For each of those elements the notion of context (C) can be used to specify a constraint, such as a group can be defined as the people located in a given room at a given time. So first we look at the mechanisms of anonymizing and blurring, which aim at restricting the data that can be accessed (D). These mechanisms aim at adding noise to the information [40]. Second, we present access control, a mechanism used to restrict the group of people accessing data (G). Third, we present labeling, a mechanism that can be used to restrict the operations that can be done with the accessed data (O). All different platforms, or middleware, provide a middleman between the mobile module and remote application module in order to ensure privacy, as we will see in the next section the architecture of this middleman can vary depending on implementation choices. Table 3.1 gives an overview of the type of mechanisms provided by existing platforms, i.e., anonymizing, blurring, access control, and labeling.

3.3.1 Restricting Data

Restricting the data that can be accessed is a way to ensure privacy in an environment where the user has little control on who will access data and for what purpose. There are two kinds of mechanisms that aim at ensuring privacy by restricting the data that can be accessed: *anonymization*, which targets the identity and *blurring*

	Anonym.	Blur.	A. Control	Label.
Confab [25]				✔
Casper [15, 35]	✔	✔		
PoRa [34]		✔		
CoBrA [12]			✔	
ContextTkt [18]			✔	
PIR [14]		✔		
Hitchhicking [45]	✔			
Mix Zones [6]	✔			
Virtual Walls [26]			✔	
PerPriv [49]	✔			
LocObf [4]		✔		
DPM [24]				✔
LanKra [28]			✔	
Mobile Gaia [13]			✔	

Table 3.1 Privacy mechanisms

which targets location. Table 3.1 presents an the privacy mechanisms supported by the surveyed above.

3.3.1.1 Anonymization

Middleware ensuring privacy through anonymizing remove the user's ID from the user's location, or other sensitive data. The idea is that the remote application will not be able to link location information with a particular user. This technique eliminates potential privacy issues, because no observer B will be able to link the information to Alice. Figure 3.2 illustrates such a mechanism and shows how the middleware replaces the identity of Alice (A) with a pseudonym (X) before sending information to the remote application. When receiving a response from the remote application it translates the pseudonym into the real identity in order to send the reply to the corresponding user. Several middleware offer such a mechanism.

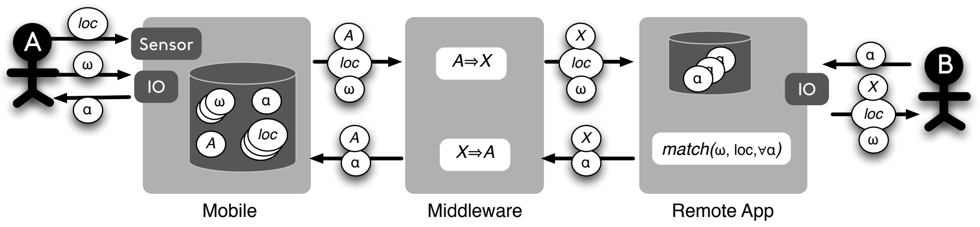


Fig. 3.2 Anonymizing

MixZone [6] provides a proxy between users and applications in order to anonymize the user's ID through pseudonyms. Pseudonyms change frequently to avoid retro-engineering of the user's ID.[4] In order to further avoid retro-engineering by tracking precise movements instead of pseudonyms special zones, called *Mix zones*, are introduced, from where no user location is sent to the remote application. These zones act like shadows in a geographical area into which users bearing certain pseudonyms enter but cannot be tracked. Therefore when they leave the zone bearing another pseudonym they are truly anonymous. Recently, such a mechanism has been adapted for the use in road networks [39] and services using non-rectangular zones to add a further security layer.

Hitchhicking [45] is another example of such a middleware and aims at providing support for application where the identity of users is not important but their location is. An example of such applications include traffic congestion applications. Since the exact location of users is important, blurring mechanisms (see below) must be avoided. Personalized Privacy Preservation (*PerPriv*) [49] is another such middleware which aims at providing anonymized access to a database. It focuses on medical records and proposes a ways to avoid aggregation of client records.

3.3.1.2 Blurring

Whereas anonymization aims at hiding identities, blurring aims at hiding location. Hiding location completely defies the purpose of location-based services, therefore mechanisms aimed at hiding location merely blur it. In the literature, several blurring techniques are used. Some of these provide less accurate location information, such as city instead of street information, others send several different locations to the server, one being the location of the requester. Figure 3.3 illustrates such a mechanism and shows how the middleware replaces Alice's location (*loc*) with an approximation ($\approx$*loc*). The remote application then returns several possible matches and the middleware chooses the most relevant one.

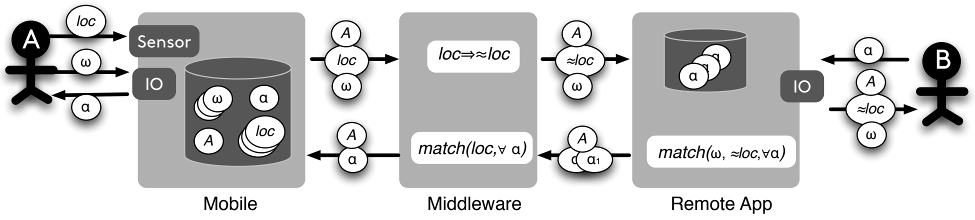


Fig. 3.3 Blurring

[4] It should be noted that simply anonymizing location data has been found to be ineffective when it comes to protecting user privacy [50].

With the *PIR-based framework*, the authors of [23] propose a way to retrieve a user's nearest neighbor (NN), or nearest point of interest (POI). In this framework the remote application first generates a map broken into different regions. The middleware then determines in which region the user is located and sends a query to the remote application for that region using a PIR protocol [14], which allows the middleware to retrieve information from a database without the server knowing which information was retrieved. Similarly, in *Casper* [15, 35] a grid is used as an overlay and all POI of on cell of the grid is retrieved in order to blur location. The aim of *Casper* is to hide a user in a crowd, and it allows to specify the minimal size of the crowd (A_{min}) and the minimum number of people in the crowd k (as in k-anonymity). Poolsappasit & Ray (*PoRa*[5]) [34] propose a system which allows to specify and enforce location privacy using different levels of granularity. The system converts physical location, i.e., geographical coordinates, into logical locations, e.g., USA, New York. Users can choose to reveal their location with more or less precision depending on the context. The authors of [4] describe several location obfuscation techniques (*locObf*). Their assumptions are that user location is a combination of a central coordinate and a radius accounting for measurement errors. The larger the radius the greater the possible measurement error. Typically, a GPS sensor provides a measurement error of about 10 meters outdoors, whereas location retrieved through tiangulation provides measurement error that can be of several hundreds of meters in a city. In their paper, they present three operators that can be used to blur the user location information in order to increase user privacy. The enlarge operator degrades de accuracy of the user's location by enlarging its radius. The shift operator shifts the center of the initial location. The reduce operator degrades the accuracy of the location by reducing its radius. Consumer typically indicate a desired location accuracy, e.g., 100 meters. These operators can be used together to further enhance privacy.

3.3.2 Restricting Access

Access control ensures that the sensitive data is only accessed by authorized applications or persons. Figure 3.4 illustrates such a mechanism and shows that the middleware checks if Bob is allowed to receive the information before it sends it to him. Different rules can be used in order to put users on the authorized list. Some of these rules can be based on identity, whereas other are based on context, such as Bob's location.

The *Context Toolkit* described in [18] provides facilities to discover peer presence and peer activity around a widget placed in a predefined location. These peers

[5] When authors do not provide a name for the middleware they describe, we provide a name based on their surnames.

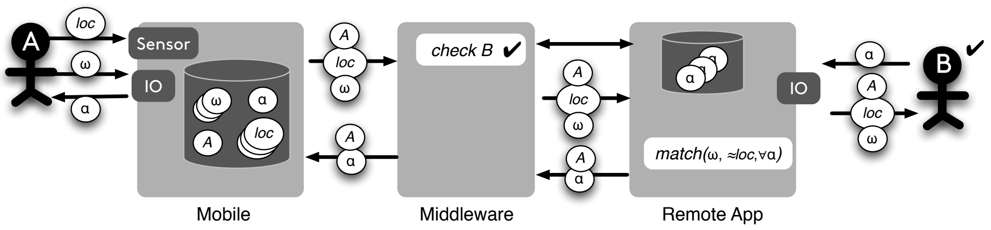


Fig. 3.4 Access Control

can then exchange information. Similarly, *CoBrA* [12] proposes the notion of *smart spaces* and their attached *context brokers*, which gather contextual information about the space and the nodes located in them. *Mobile Gaia* [13] is another such middleware and allows devices to join a *personal space*. Once such a personal space is set up, a discovery service invites nearby devices to join the space and share resources and services. A location service is then responsible for location-awareness in the space. Access control identifies what information or resource is shared with a specific space name. Virtual Walls [26] allow users to set barriers around a physical space within which private information called about them (also called footprints) can be gathered. The idea of virtual walls is to provide a easy-to-use abstraction to set privacy policy through a GUI and the wall analogy. It provides three levels of privacy for each space; transparent, which lets outsiders access information about location and identity of peers inside the space, translucent, which anonymizes the peers inside the space, and opaque, which prohibits outsiders from accessing data inside the space. In [37], the authors propose a proximity-based detection service leveraging on existing social network links (such as being friends on Facebook) in order to share secret keys. User identity is coded through these keys and only revealed when two users are within some defined geographic proximity.

3.3.3 Restricting Operations

Some middleware systems provide means to control usage of data. These middleware use a mechanism known as *labeling*. The idea behind this mechanism is to specify what kind of operations are not allowed for a certain data and the system will ensure that such operations will not be executed. Figure 3.5 illustrates such a mechanism and shows how the middleware encapsulates the data with a label, then the data is sent to a trusted part of the remote application where the user can only access the data as specified by the label.

Confab [25] offer such a labeling mechanism. Each user has a confined environment called an *infospace* where data is stored and privacy policies are enforced. A user can decide on the lifetime of data or if the data can be forwarded. When data is sent

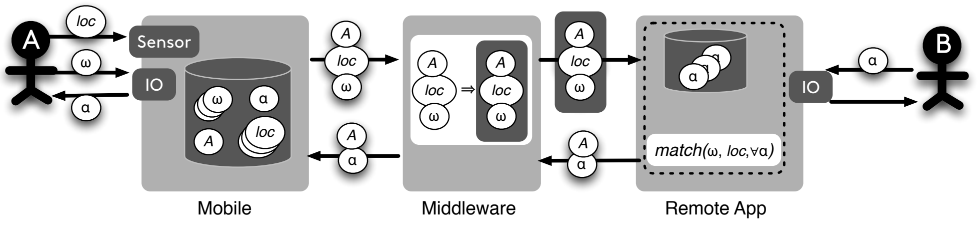


Fig. 3.5 Labeling

to another user, it is restricted to the infospace, which enforces the privacy policies. *Confab* also allows to control access to the data as well as the granularity of the data and allows to tag the data in order to track it for digital rights management (DRM) purposes. The dynamic privacy management (*DPM*) [24] is based on the P3P internet privacy protocol. In this protocol (P3P), before a user accesses a web page, the page sends its privacy policy, that is, it sends a file stating what kind of information will be retrieved through the site and for what purpose. Then the browser compares this privacy policy with user preferences and if there is a match the page is displayed, if not, the user is notified [16]. DPM uses the same XML syntax as P3P but extends it to allow applications to specify preferences related to time, location and the identity of the data owner. The idea is that a mechanism announces and describes required personal data, the scope, the intended use an the consequence. User agents have an embedded trust engine that does the matching between preferences and proposals. Unlike CONFAB, where policies are enforced by the system, DPM allows or refuses access based on the advertised use beforehand. Similarly Langendoerfer & Kraemer (*LanKra*) [28] present a service with a privacy match based on user preferences and service policies. Their protocol is implemented over a security layer which aims at restricting eavesdropping on communications between server and client.

3.4 Middleware Architecture

We use the architecture dimension to measure the ability of existing middleware solutions to support heterogeneous computing environments. We classify these solutions along two architectural dimensions, namely decentralization and portability. Decentralization measures a platform's dependence on specific components. Intuitively, a centralized middleware, i.e., one that relies on some central entity, is more vulnerable and less flexible than a decentralized one, which does not depend on any fixed component. It should be noted that by decentralization we do not imply that the whole system is decentralized, but that the middleware does not include a centralized entity, even if it accesses information on a centralized server, typically through a web gateway. Portability classifies platforms in two groups: portable platforms, can run on many different operating systems, and operating system dependent platforms, which can only run on one or a few operating systems.

3.4.1 Decentralization

The decentralization of a middleware solution indicates its resilience to network topology changes. The more a middleware is decentralized, the more it is resilient to such changes. Centralized solutions tend to make strong assumptions about the network topology, as they often rely on some fixed network infrastructure or at least on the central role played by some predefined components. As a consequence, these solutions may stop working properly as soon as the connection to that central component is lost. Decentralized platforms, on the other hand, weaken these assumptions, by considering a network of peers that can equally contribute to the accomplishment of the tasks. Decentralized architectures are usually adopted when designing a middleware to be deployed in a ad hoc environment, where communication occurs spontaneously and in a peer-to-peer fashion. Table 3.2 presents an overview of the architectures of the surveyed middleware.

	Decentralized	Portable
Confab	✔	✔
Casper		
PoRa		✔
CoBrA		✔
ContextToolkit		
PIR Framework	✔	
Hitchhicking	✔	✔
Mix Zones		
Virtual Walls		
PerPriv		
LocObf		
DPM		✔
LanKra		
Mobile Gaia		

Table 3.2 Middleware architecture

3.4.1.1 Centralized Solutions

An architecture with a centralized trusted component, referred to as a trusted third party (TTP), strongly facilitates the implementation of privacy policies. Such a TTP has typically access to privacy sensitive information from different clients and to the remote application and can make sure that none of the privacy sensitive material leaks to unauthorized persons or applications. This approach has been adopted by many middleware solutions with different names for the centralized component. *Casper* uses its location anonymizer as a TTP in order to hide the user's identity and location from the location-based database. *PoRa* uses a trusted remote service provider which accesses several databases through a privacy agent. *CoBrA* uses a centralized context broker with a privacy management module as a proxy between the mobile application and other data. *ContextToolkit* has a central privacy broker which can be trusted. *Virtual Walls* trust a centralized context server. *DPM* uses a privacy data mediator as a TTP between the user and the merchant. *LanKra* propose a distributed privacy layer on both client and server with a security layer around.

3.4.1.2 Decentralized Solutions

Several middleware are built in a decentralized fashion and avoid the usage of a TTP. Hitchhiking trust only the client device. It removes the identity of the client in all client server interactions. In the PIR Framework, the nearest neighbor is returned with no need for a trusted third party. The middleware sends a request to the location-based service (LBS) for points of interests (POI) located within a cloaked area, then the LBS returns all the POI in this area, then the middleware selects the closest POI and returns it to the application. *Confab* relies on trusted entities called *infospaces*. Infospaces data can be stored and its access is controlled, data can then be removed after a desired time to live. Data cannot be pushed beyond the reach of *Confab* otherwise the labeling mechanism cannot enforce its rules. These infospaces can be centralized but also decentralized and distributed on users' devices. For other middleware solutions, the architecture is, to the best of our knowledge, not explicitly described. These solutions include *MixZones*, *PerPriv* and *locObf*.

3.4.2 Portability

Platforms implemented in a portable programming language such as Java[6] can more easily be deployed on heterogenous operating systems. The following middleware solutions have adopted this approach: *Confab*, *PoRa*, *CoBra*, *Hitchhiking*, and *DPM*.

[6] http://java.sun.com

3.5 Discussion

Several middleware have been proposed to address privacy issues for context-aware mobile applications. First we presented four different privacy mechanisms, i.e., data anonymizing, location blurring, access control and labeling. Then, we classified these solutions a depending on their implementation, whether they are decentralized or wether they rely on a TTP, we consider the TTP to be the default setting. We also investigated the portability of the solutions in whether they are implemented in Java or not.

3.5.1 Overview

We categorized 14 platforms summarized in Table 3.1 and Table 3.2 gives an overview of this classification. Five are centered around an access control mechanism, four around a blurring mechanism, four around anonymizing and two around labeling. About a third (5/14) of the platforms are portable and only 3 out of 14 are decentralized. Also, there are no decentralized platform providing access control or labeling.

3.5.2 Choosing an Adequate Middleware

In order to choose an adequate middleware for the development of an application, several features should be considered, the most important being the privacy mechanism provided. As we shall see the application specifications will determine what kind of mechanism is relevant. Then, the intended deployment environment of the application, will put constraints on the middleware architecture and implementation that should be considered. Hereafter we focus on the decision regarding the privacy mechanisms. Table 3.3 shows which mechanisms may be used in a specific type of location-aware application. The *labeling* and *access control* mechanisms may be useful in every type of application. *Anonymizing* and *blurring* may not be always useful depending on the location-aware application type. In *localized ads* or *geo-information* application, like virtual tourism guide, both anonymizing and blurring may be used: we can provide the content to an anonymous user and the content is related to a zone and not a specific point.

In *friend finder* applications or other king of *radar* applications (generally related to social networks), because users identities are the main concern, only blurring mechanisms may be used to ensure users privacy. In *crowd information* or *traffic information* application like traffic jam analyzer or crowd detectors, the identity of the users is not important. But such applications need precise user locations

	Anonym.	Blur.	Acc. control	Label.
LAD	✔	✔	✔	✔
Friend finder		✔	✔	✔
Traffic info	✔		✔	✔
LB games			✔	✔

Table 3.3 Privacy mechanisms and application types

to determine the density in a given area. So blurring is not adequate for these applications. In location-based games, both users identities and location accuracy are needed because the application needs to match a determined player with its accurate location in the game field.

3.6 Conclusion

Building appropriate privacy sensitive middleware is a challenging endeavor, especially for context-aware applications where mobility increases on one hand the service possibilities, i.e., location-based services, but on the other hand also increases the dissemination of personal and sensitive data. In this paper we specified the problem of privacy and the difference between privacy and security. Then we surveyed existing middleware solution along two dimensions, namely their approach towards privacy mechanisms and their architecture. Our survey shows that efforts are undertaken to provide support for privacy for context-aware applications. However there is still room for improvement. Among these improvements one can mention the following three, which can be used as starting points for future research: (1) A platform providing several privacy mechanisms according to some application scenarios. (2) An increase in decentralized architecture. Most middleware described in the surveyed articles rely on a centralized broker, which can be limiting for mobile context-aware applications. especially in ad hoc settings. (3) An increase in trust limitation. All surveyed middleware seem to trust the application on the user's device. However there is no guarantee that this application will not send private information out through a backdoor once it was safely retrieved from the remote application. Investigating this issue therefore represents an important research avenue.

References

[1] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: a mobile context-aware tour guide. *Wirel. Netw.*, 3(5):421–433, 1997.

[2] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC'99)*, pages Pages 304–307, London, UK, 1999. Springer-Verlag.

[3] Mark S. Ackerman. Privacy in pervasive environments: next generation labeling protocols. *Personal Ubiquitous Comput.*, 8(6):430–439, 2004.

[4] Claudio A. Ardagna, Marco Cremonini, Sabrina De Capitani di Vimercati, and Pierangela Samarati. An obfuscation-based approach for protecting location privacy. *IEEE Transactions on Dependable and Secure Computing*, 99(1), 2007.

[5] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, Vol. 2(No. 4):Pages 263–277, June 2007.

[6] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[7] Staffan Björk, Jennica Falk, Rebecca Hansson, and Peter Ljungstrand. Pirates! using the physical world as a game board. In *Proceedings of the Human-Computer Interaction Conference(Interact'01)*. IOS Press, 2001.

[8] Susanne Boll, Jens Krösche, and Christian Wegener. Paper chase revisited: a real world game meets hypermedia. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 126–127, New York, NY, USA, 2003. ACM.

[9] P. J. Brown. The stick-e document: a framework for creating context-aware applications. In *Proceedings of EP'96, Palo Alto*, volume Vol. 8, pages Pages 259–272. Electronic Publisher, January 1996.

[10] David Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[11] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

[12] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. *Knowl. Eng. Rev.*, Vol. 18(No. 3):Pages 197–207, September 2003.

[13] S. Chetan, J. Al-Muhtadi, R. Campbell, and M.D. Mickunas. Mobile gaia: a middleware for ad-hoc pervasive computing. In *Proceedings of Consumer Communications and Networking Conference (CCNC'05)*, pages pages 223–228. IEEE Computer Society, January 2005.

[14] BENNY CHOR, ODED GOLDREICH, EYAL KUSHILEVITZ, and MADHU SUDAN. Private information retrieval. *Journal of the ACM*, 45(6):965–982, November 1998.

[15] Chi-Yin Chow, Mohamed F. Mokbel, and Tian He. Tinycasper: a privacy-preserving aggregate location monitoring system in wireless sensor networks. In

SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pages 1307–1310, New York, NY, USA, 2008. ACM.

[16] Lorrie Faith Cranor and Lawrence Lessig. *Web Privacy with P3P.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[17] MJ Culnan. Protecting privacy online: is self-regulation working? In *In Journal of Public Policy Market*, pages 20–26, 2000.

[18] Oleg Davidyuk, Jukka Riekki, Ville-Mikko Rautio, and Junzhao Sun. Context-aware middleware for mobile multimedia applications. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia (MUM '04)*, pages Pages 213–220, New York, NY, USA, 2004. ACM.

[19] Anind K. Dey, Gregory D. Abowd, and Andrew Wood. Cyberdesk: A framework for providing self-integrating context-aware services. In *Proceedings of the Symposium on User Interface Software and Technology*, pages 75–76. ACM, 1997.

[20] Patrick Th. Eugster, Benoît Garbinato, and Adrian Holzer. Middleware support for context-aware applications. *Chapter 14 of Middleware for Network Eccentric Applications. Springer*, 2009.

[21] Martin Flintham, Steve Benford, Rob Anastasi, Terry Hemmings, Andy Crabtree, Chris Greenhalgh, Nick Tandavanitj, Matt Adams, and Ju Row-Farr. Where on-line meets on the streets: experiences with mobile mixed reality games. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 569–576, New York, NY, USA, 2003. ACM.

[22] Fabien L. Gandon and Norman M. Sadeh. Context-awareness, privacy and mobile access: a web semantic and multiagent approach. In *Proceedings of the 1st French-speaking conference on Mobility and ubiquity computing (UbiMob'04)*, pages 123–130, New York, NY, USA, 2004. ACM.

[23] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 121–132, New York, NY, USA, 2008. ACM.

[24] Dan Hong, Mingxuan Yuan, and Vincent Y. Shen. Dynamic privacy management: a plug-in service for the middleware in pervasive computing. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, pages 1–8, New York, NY, USA, 2005. ACM.

[25] Jason I. Hong and James A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 177–189, New York, NY, USA, 2004. ACM.

[26] Apu Kapadia, Tristan Henderson, Jeffrey Fielding, and David Kotz. Virtual walls: Protecting digital privacy in pervasive environments. In Anthony

LaMarca, Marc Langheinrich, and Khai Truong, editors, *Pervasive Computing*, volume 4480 of *Lecture Notes in Computer Science*, pages 162–179. Springer Berlin / Heidelberg, 2007.

[27] Kristian Ellebaek Kjaer. A survey of context-aware middleware. In *Proceedings of the 25th conference on IASTED International Multi-Conference (SE'07)*, pages Pages 148–155, Anaheim, CA, USA, 2007. ACTA Press.

[28] Kraemer. Towards user defined privacy in location-aware platforms. In *Proceeding of the 3rd international Conference on Internet computing, USA. CSREA Press*, 2002.

[29] John Krumm. A survey of computational location privacy. *Personal Ubiquitous Comput.*, 13:391–399, August 2009.

[30] Marc Langheinrich. A privacy awareness system for ubiquitous computing environments. In *UbiComp '02: Proceedings of the 4th international conference on Ubiquitous Computing*, pages 237–245, London, UK, 2002. Springer-Verlag.

[31] Carsten Magerkurth, Adrian David Cheok, Regan L. Mandryk, and Trond Nilsen. Pervasive games: bringing computer entertainment back to the real world. *Comput. Entertain.*, 3(3):4–4, 2005.

[32] John Mathew, Suprateek Sarker, and Upkar Varshney. M-commerce services: Promises and challenges. *Communications of the AIS (CAIS)*, Vol. 14(Article 25), 2004.

[33] Kevin D. Mitnick and William L. Simon. *The Art of Deception: Controlling the Human Element of Security.* John Wiley & Sons, Inc., New York, NY, USA, 2003.

[34] Mohamed F. Mokbel. Towards privacy-aware location-based database servers. *Data Engineering Workshops, 22nd International Conference on*, 0:93, 2006.

[35] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. The new casper: query processing for location services without compromising privacy. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[36] Ginger Myles, Adrian Friday, and Nigel Davies. Preserving privacy in environments with location-based applications. *IEEE Pervasive Computing*, 2(1):56–64, 2003.

[37] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, and Dan Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.

[38] United Nations. Universal declaration of human rights. In *General Assembly Resolution 217 A(III)*, 1948.

[39] Balaji Palanisamy and Ling Liu. Mobimix: Protecting location privacy with mix-zones over road networks. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 494–505, Washington, DC, USA, 2011. IEEE Computer Society.

[40] Blaine A. Price, Karim Adam, and Bashar Nuseibeh. Keeping ubiquitous computing to yourself: a practical model for user control of privacy. *Int. J. Hum.-Comput. Stud.*, 63(1-2):228–253, 2005.

[41] Omer Rashid, Ian Mullins, Paul Coulton, and Reuben Edwards. Extending cyberspace: location based games using cellular phones. *Comput. Entertain.*, 4(1):4, 2006.

[42] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, pages Pages 89–101, Santa Cruz, CA, US, 1994. IEEE Computer Society.

[43] Michael Spreitzer and Marvin Theimer. Scalable, secure, mobile computing with location information. *Commun. ACM*, 36(7):27, 1993.

[44] Mike Spreitzer and Marvin Theimer. Providing location information in a ubiquitous computing environment (panel session). *SIGOPS Oper. Syst. Rev.*, 27(5):270–283, 1993.

[45] Karen P. Tang, Pedram Keyani, James Fogarty, and Jason I. Hong. Putting people in their place: an anonymous and privacy-sensitive approach to collecting sensed data in location-based applications. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 93–102, New York, NY, USA, 2006. ACM Press.

[46] Daniel Terdiman. Making wireless roaming fun. Wired News Online, April 2003.

[47] Yanna Vogiazou, Bas Raijmakers, Ben Clayton, Marc Eisenstadt, Erik Geelhoed, Jon Linney, Kevin Quick, Josephine Reid, and Peter Scott. You got tagged!: The city as a playground. In *Second International Conference on Appliance Design (2AD'04)*, Bristol, UK, May 2004.

[48] Skyhook Wireless. Location aware app report. In *http://www.locationrevolution.com/stats/skyhookjulyreport.pdf.* Skyhook Wireless, 2009.

[49] Xiaokui Xiao and Yufei Tao. Personalized privacy preservation. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 229–240, New York, NY, USA, 2006. ACM.

[50] Hui Zang and Jean Bolot. Anonymization of location data does not work: a large-scale measurement study. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, MobiCom '11, pages 145–156, New York, NY, USA, 2011. ACM.

[51] Yi Zheng, Dickson K. W. Chiu, Hongbing Wang, and Patrick C. K. Hung. Towards a privacy policy enforcement middleware with location intelligence. In *EDOCW '07: Proceedings of the 2007 Eleventh International IEEE EDOC Conference Workshop*, pages 97–104, Washington, DC, USA, 2007. IEEE Computer Society.

Part II
Context-aware Dissemination Algorithms

Chapter 4
Six-shot Broadcast: a context-aware algorithm for efficient message diffusion in MANETs

Published in the proceedings of On the Move to Meaningful Internet Systems: OTM 2008
OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part I.
Pages 625-638
Publisher: Springer
ISBN: 978-3-540-88870-3

Abstract In this paper, we introduce *six-shot broadcast (6SB)*, a new context-aware message diffusion algorithm that uses location information to fine-tune its broadcasting process. Message diffusion is indeed one of the core challenges brought up by distributed systems and has therefore largely been studied in the context of traditional network structures such as the Internet. With the emergence of mobile ad hoc networks *(MANETs)*, new broadcasting algorithm especially geared at these networks have been introduced. These algorithms must reach two conflicting objectives when broadcasting a message, namely reliability vs. efficiency. That is, they must maximize the number of nodes that deliver the message (reliability), while minimizing the number of nodes that forward the message (efficiency). In recent years as more and more mobile devices have become context-aware, several broadcasting algorithms have been introduced using contextual information, such as location, in order to increase reliability and efficiency. Along that line, we provide a in-depth performance evaluation of our 6SB algorithm, by comparing it to similar broadcasting algorithms also targeted at *MANETs*. Our results show that *6SB* competes with the most efficient algorithms in high densities of nodes and offers increased reliability in low densities at a reasonable overhead.

4.1 Introduction

During the past years we have been witnessing a massive increase of mobile devices. These devices are now ubiquitous and changed the traditional distributed systems from centralized and wired architectures to dynamic, heterogeneous and frequently changing network architectures like mobile ad hoc networks *(MANETs)*. These architectures offer many new opportunities for application developers and new challenges for networking protocol designers. The former develop new types of dynamic mobile applications used for example in traffic jam prevention, information dissemination in crowds, strategic data gathering in hostile environments, or peer-to-peer mobile games. The later focus on designing low-level protocols such as broadcast, consensus or atomic commit to create the building blocks for the application developers. In this paper, we introduce *six-shot broadcast (6SB)* as one of these building blocks.

4.1.1 Mobile ad hoc networks

Mobile ad hoc networks *(MANETs)* are networks of mobile devices without fixed infrastructure. In such a network, every node is directly connected to all nodes located within the range of its radio signal, also referred to as the *technical range*. These nodes are called *neighbors*. Since nodes are mobile, neighborhoods change over time as nodes get in and out of each others technical range.

Communication between neighbors is trivial, as every message emitted by a node is received by all neighbors, thus no routing is necessary. Communication with nodes located outside the neighborhood is a more interesting aspect of *MANETs*, since it implies *multi-hop* message diffusion, where the message must be forwarded by one or more intermediate relays between the sender and the receiver. Henceforth, we will only consider multi-hop message diffusion.

4.1.2 Message diffusion in MANETs

In this paper, we address one kind of message diffusion, namely broadcast, where one node sends a message to all nodes in the network. Broadcasting in *MANETs* is made difficult by three of its inherent properties: (1) frequent network configuration change due to the mobility of devices, (2) decentralized architecture due to the absence of fixed infrastructure, and (3) limited resources due to small mobile devices. Such resources include battery and CPU power and the limited network bandwidth, which favors message collisions and might lead to what is called a *broadcast storm* [6] paralyzing the network. In order to evaluate and compare different broadcasting algorithms, two dimensions can be considered:

Reliability indicates the number of nodes in the network that deliver a message compared to the number of nodes that should have delivered the message. We measure this dimension via the *delivery ratio*. The higher the delivery ratio, the higher the reliability.

Efficiency indicates the cost of broadcasting in terms of message retransmissions. This dimension is measured by the *forward ratio*, the number of nodes relaying a message divided by the total number of nodes in the network. The higher the forwarding ratio, the poorer the efficiency.

The challenge of broadcasting in *MANETs* is to ensure that a high number of nodes deliver the message (reliability), while a small number of nodes retransmit this message (efficiency).

4.1.3 Contribution and roadmap

In Section 4.2, we present a novel context-aware broadcasting algorithm called *six-shot broadcast (6SB)* which constitutes the main contribution of this paper. Section 4.3 then discusses related work before Section 4.4 presents the second contribution of the paper, a thorough performance evaluation and comparison of *6SB* and related algorithms. Section 4.5 wraps up this paper with concluding remarks and hints on future research opportunities.

4.2 Six-shot Broadcast

The *six-shot broadcast (6SB)* algorithm is a *context-aware* broadcasting algorithm. which uses location as context information. The *6SB* interface offers the two following typical primitives:

6SB-BROADCAST(m): broadcasts the message m to all nodes located in the network.
6SB-DELIVER(m): acts as a callback when message m is received.

The particularity of *6SB* resides in the scheme it uses to decide whether or not to forward a given message. This scheme is based on a widespread mechanism that we dub *wait and count*. With this mechanism, when a message m is received, a node initiates a waiting time during which it counts retransmissions of m. When the waiting time elapses m is forwarded unless the number of retransmissions is greater than a predetermined *threshold*. *6SB* assigns a different waiting time depending on the geographical location of nodes. The idea is that before a node sends a message m, it associates six geographical *targets* to it as depicted in Figure 4.1 with Alice as the sender. The number six is chosen because of its interesting property ensuring that every node in the sender's neighborhood can be reached by at least two targets. Among all nodes that receive m, only those located closest to a target should forward m. Note that only nodes located in what is called the *forward zone* can possibly forward messages, all nodes located in the *no forward zone* never forward a message.

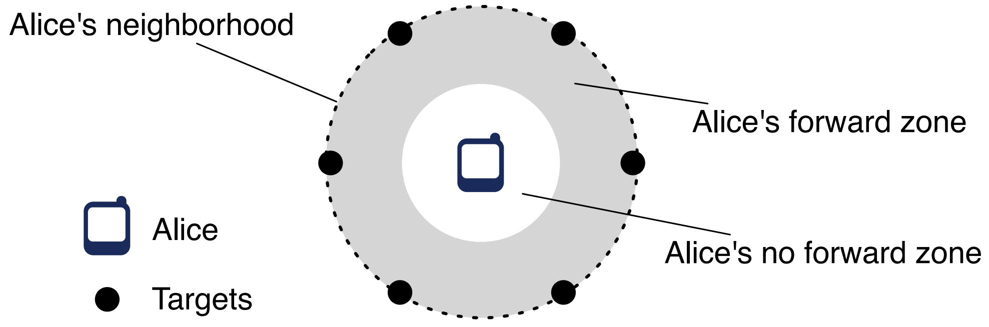


Fig. 4.1 6SB principle

4.2.1 Underlying services

6SB is built as a primitive that can be used by the application layer and uses two underlying services. As it is location-based it uses an underlying *Location Service (LS)* and like many routing algorithms in *MANETs* it is built upon a *MAC* data link layer.

Location Service. This service allows to access the node's geographical location via a location sensor such as a GPS or location beacons. The location service offers the following two primitives:

LS-GETPOSITION(): retrieves the node's current geographical location.
LS-GETDISTANCE(l_1, l_2): returns the distance between locations l_1 and l_2.

MAC layer. In *MANETs*, the *MAC* layer is used to broadcast and receive messages in a node's neighborhood. We model this communication capability via the two following primitives:

MAC-BROADCAST(m): broadcasts the message m to all nodes located in the sender's neighborhood.
MAC-DELIVER(m): acts as a callback when message m is received.

4.2.2 Implementation

At the heart of *6SB* lay the 6SB-BROADCAST and the 6SB-DELIVER primitives presented previously, as well as two important functions, namely GETTARGETS, and GETDELAY. The detail of the *6SB* algorithm is presented in Algorithm 4.8.

The initialization phase. *6SB* uses three global variables: the *threshold* indicates the number of received retransmissions after which a message does not need to be

```
1: uses MAC, Location Service

2: INIT
3:     $threshold \leftarrow ...$                    {threshold fixed by the user}
4:     $nfz \leftarrow ...$                          {size of the no forward zone fixed by the user}
5:     $counter \leftarrow \langle 0, 0, ..., 0 \rangle$    {list of counters}

6: To execute 6SB-BROADCAST($m$) :
7:     $m.center \leftarrow$ LS-GETPOSITION          {sets the center of m to the postion of the sender}
8:     MAC-BROADCAST($m$)                            {broadcasts m}

9: 6SB-DELIVER($m$) occurs as follows:
10:     upon MAC-DELIVER($m$) do                     {when a message is delivered by the MAC layer}
11:         if $counter[m] = 0$ then                 {if m is received for the first time}
12:             6SB-DELIVER($m$)                     {delivers m, ¬ blocking}
13:             $counter[m] \leftarrow 1$            {initializes the message counter for m}
14:             $delay \leftarrow$ GETDELAY($m$)     {sets the waiting delay}
15:             if $delay \geq 0$ then               {if the delay is valid}
16:                 WAIT($delay$)                    {waits until the delay elapses}
17:                 if $counter[m] \leq threshold$ then    {if the threshold is not reached}
18:                     6SB-BROADCAST($m$)           {forwards m}
19:         else
20:             $counter[m] \leftarrow counter[m] + 1$     {increments the msg counter}

21: function GETDELAY($m$) :
22:     $myPosition \leftarrow$ LS-GETPOSITION       {gets the current position of the node}
23:     if (LS-GETDISTANCE($m.center, myPosition$) $< nfz$) then
24:         return $-1$                              {returns -1 if the node is in the no forward zone}
25:     else
26:         $distance \leftarrow -1$
27:         for all $t \in$ GETTARGETS($m.center$) do     {gets the nearest target}
28:             if (LS-GETDISTANCE($myPosition, t$) $< distance$) $\vee$ ($distance = -1$) then
29:                 $distance \leftarrow$ LS-GETDISTANCE($myPosition, t$)
30:     return DELAY($distance$)                     {returns the delay in function of the distance}

31:     function GETTARGETS($center$) :
32:         $targets \leftarrow \emptyset$
33:         for $0 <= i < 6$ do                      {creates six targets}
34:             $t_i \leftarrow \langle center.x + sin(30 + 60 \times i) \times range, center.y + cos(30 + 60 \times i) \times range \rangle$
35:             $targets \leftarrow targets \cup \{t_i\}$
36:         return $targets$                         {returns the targets}
```

Algorithm 4.8 Six-shot broadcast algorithm

forwarded anymore. The nfz variable indicates the size of the no-forward zone, and the message *counter* list, which is indexed by message IDs and keeps track of the number of received retransmissions (lines 2-5).

The broadcast primitive. When the 6SB-BROADCAST(m) primitive is called with a message m as parameter, a broadcast is initiated. This process adds the location of the sender to the message m before it is broadcasted to the neighborhood via the MAC-BROADCAST primitive. The sender's location is obtained via the location service's LS-GETPOSITION primitive (lines 6-8).

The delivery primitive. When the *MAC* layer receives a message m for the first time through the MAC-DELIVER callback , the 6SB-DELIVER callback is triggered and the counter for m is set to 1. Then the node waits for a delay determined by the GETDELAY function if the delay is valid. During this waiting time, when other copies of m are received, they increment the message *counter*. After the waiting time elapses, m is forwarded using the 6SB-BROADCAST primitive if its *counter* is less or equal to the *threshold* (lines 9-15).

The delay computing function. This function is central to the *6SB* algorithm. When GETDELAY is called, it checks if the node is in the no forward zone. If it is, -1 is returned, otherwise the six targets are retrieved by the GETTARGETS function. Then the DELAY() function computes the node's waiting delay proportionally to its distance to the closest target. We encapsulated this last function since it must be fine tuned according to the application context and the technology used (lines 35-36).

The target computing function. The six targets are computed via the GETTARGETS function by calculating the coordinates of each target based on the location of the sender (the center) and the technical range (lines 8-14).

4.3 Related work

Many different types of algorithms for broadcasting in *MANETs* exist [3, 4, 5, 6, 1, 8]. In the following, we investigate the algorithms closest to *6SB*. We first present the *counter-based scheme*, which created the foundation of the *wait and count* mechanism used in *6SB*. Then we investigate four context-aware algorithms, namely the *power-aware message propagation algorithm* and the *optimized flooding protocol*, which fine tune waiting delays according specific locations, similarly to *6SB* and the *location-based scheme* as well as the *area-based beaconless algorithm*, which use location information to fine tune the counting rules.

4.3.1 Counter-based scheme

In the *counter-based scheme (CBS)* [6], when a node receives a new message m, it fixes a random waiting delay before making the forwarding decision. During this delay, the node counts the number of retransmission of m it receives. After the waiting delay has elapsed, the message is only forwarded if the number of retransmissions is less than a predetermined *threshold*. As a result, the forwarding ratio depends on the node density, i.e., in low density areas this ratio will be high, whereas in high density areas it will be very low. Simulation results presented in Section 4.4 show that this simple algorithm turns out to be very competitive as it does not concede much reliability for highly improved efficiency.

The hop-count aided broadcasting algorithm (*HCAB*) presented in [4] is similar to CBS except that it fine-tunes the counting rule. Messages in *HCAB* store the number of hops they perform in a variable that we call *hopNb*. When a node receives a message for the first time it gets the value v of the message's *hopNb*, sets a random waiting time and then *only* counts retransmissions of the message with *hopNb* greater than v. Then similarly to *CBS* when the waiting time elapses, the message is forwarded if the retransmission counter is below the *threshold*.

4.3.2 Power-aware message propagation algorithm

The *power-aware message propagation algorithm (PAMPA)* [5] has the exact same mechanism as *CBS* except that the delay is not fixed randomly but it is based on the intensity of the signal at which a message is received. The stronger the signal, the longer the delay. The idea behind *PAMPA* is that forwarders should be located as far away as possible from the source node. With its mechanisms, *PAMPA* ensures that only neighbors located on the outskirts of a node's neighborhood will forward messages.

4.3.3 Optimized flooding protocol

The *optimized flooding protocol (OFP)* [8] is a context-aware broadcasting algorithm which aims at providing a highly efficient message forwarding scheme. *OFP* transposes the broadcasting problem in MANETs into the following geometric optimization problem: *how can we minimize the number of circles it takes to cover a certain surface under the constraint that every circle must have its center on the area covered by another circle?* The answer to this problem is to divide the surface into hexagons the size of a circle, and then to draw a cercle on each summit. Transposed back to broadcasting in MANETs, the center of each cercle represents the ideal location for forwarders. Figure 4.2 depicts this idea with a central source node broadcasting a message, which will first be forwarded by three first-hop forwarders and then by six second-hop forwarders and so forth. This solution allows to have the absolute minimum number of forwarders in order to reach all nodes within a surface in an ideal case.

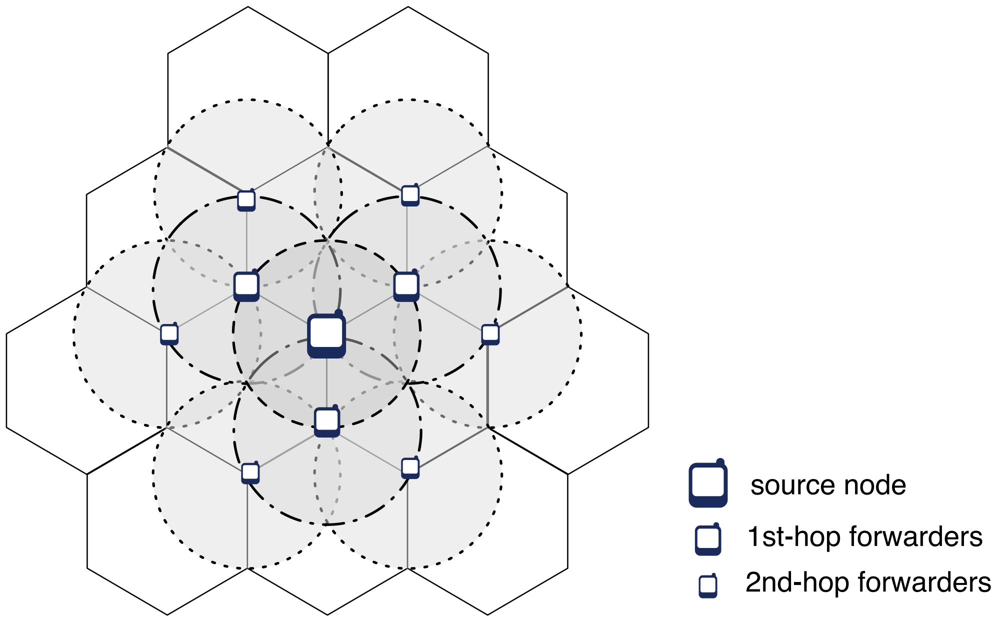


Fig. 4.2 OFP principle

The *OFP* algorithm works also like *CBS* except that the waiting time is set according to the distance of each node with the closest ideal forwarding spot. The closer to the spot, the shorter the waiting time.

4.3.4 Area-based algorithms

In the previously presented context-aware algorithms, context is used to fine-tune the waiting time. In other algorithms, such as the *location-based scheme (LBS)* presented in [6] and the *area-based beaconless algorithm*, abreviated *ABBA* [1], primarily use context to fine-tune the counting rule. In *ABBA* and *LBS* when a message is received for the first time, a waiting time is set either randomly or according to the proximity of the sender very much like previously presented algorithms. The major particularity of *ABBA* and *LBS* resides within their counting rule. Instead of counting the number of retransmissions received, nodes use the location of senders to determine what proportion of their neighborhood is covered by the senders' neighborhoods. If the proportion is less than a threshold value, the message is forwarded.

4.3.5 Summary

All algorithms presented above use the *wait and count* mechanism introduced by *CBS*. *OFP*, *PAMPA* are the closest efforts to *6SB* in that they also add a context-based scheme in order to delegate the forwarding task to nodes located nearby strategic positions. Each of these algorithms defines these positions differently. *PAMPA* selects nodes located closest to the limit of its neighborhood, *OFP* selects nodes closest to three targets, and *6SB* selects node closest to six targets. Figure 4.3 illustrates each algorithm's forwarding pattern.

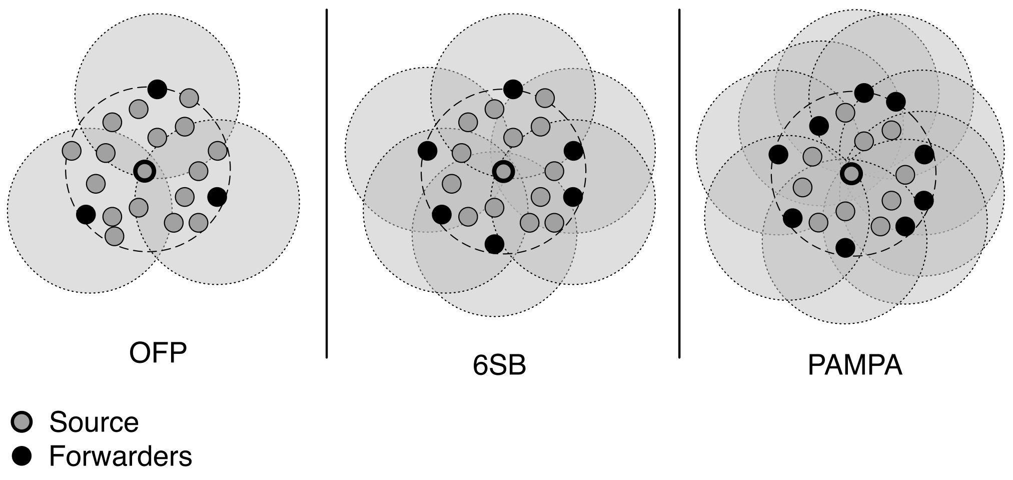


Fig. 4.3 forwarding patterns

The main difference between these algorithms in terms of forwarding pattern is the number of potential forwarders used. Pampa has the highest number and OFP the lowest number. In the following section, we compare the performance of these algorithms in order to measure the difference in efficiency and reliability that this difference implies.[1]

4.4 Performance evaluation

We measure performance in terms of forward and delivery ratios, and compare them to the algorithms presented in the related work (Section 4.3). Our results change depending on the density of nodes in the field. In high densities, *6SB* performs as well as *OFP*, which is the best algorithm. In low densities, *6SB* performs better

[1] This difference also implies variations in message diffusion speed, which we do not analyse in this paper.

than *OFP* in term of reliability for small overhead in terms of efficiency. Our results also demonstrate that minor mobility has no influence on the performance results. Before discussing these results in detail, we present the simulation settings used for the evaluation.

4.4.1 Simulations environment

To compute the delivery and forward ratios of the algorithms, we implemented them using the Java-based Sinalgo framework [7] and ran extensive simulations. The context we simulate is one of an event, such as a conference or a festival, where every participant has a mobile device connected to a MANET with IEEE 802.11n WiFi links and a GPS chip. An example of such a device is Apple's iPhone or Nokia's E71. In our context, users are static or moving around at walking speed.

Density and connection degree. We define three parameters to characterize the density and the degree of connections of the network: the *map size*, the *technical range* and the *number of nodes*. We choose a square map of 200 meters width with a technical range of 20 meters. [2] In every simulation, one node broadcasts a message. Each simulation was run 100 times in an initially fully connected network of 160 up to 2000 nodes. This means that every node has an from around 4 up to 60 neighbors. The map shape we use for the simulations is a torus to avoid having to deal with special conditions at the field limits. Table 4.1 summarizes the general simulations parameters we use.

Parameters	Value
Map size	200 $m \times$ 200 m
Map shape	torus
Map area	40000 m^2
Technical range	20 m
Number of node	$160 - 2000$
Number of sender	1

Table 4.1 General simulations parameters

Mobility. There exist several mobility models and many studies about their realism in MANETs. One of the most used is the Random Waypoint Model [2]. Our torus-shaped map resolves a known issue of the Random Waypoint Model, i. e., the

[2] Note that IEEE 802.11n allows outdoor communications up to 140 meters between laptops and wireless routers and 70 meters for indoor communication. After testing the range of various smart phones, we found values between 20 and 30 meters to be the most reasonable.

fact that node tend to converge to the center of the simulation field. In our context, we defined a scenario with mobility at walking speed. So, we set the speed of our Random Waypoint model uniformly distributed between 1 and 2 meters per second. In this mobility model, nodes choose a random waypoint on the field towards which they move. When the waypoint is reached, node wait for a random uniformly distributed random delay from 0 to 10 seconds before choosing another waypoint. The message transmission speed used in the simulation is 0.1 second and represents the time between a MAC-BROADCAST and a MAC-DELIVER in a 20 meter range. Table 4.2 summarizes the mobility and time parameters used.

Parameters	Value
Mobility	Random Waypoint in a torus
Delay	Uniform 0 - 10 s
Nodes speed	Uniform 1 - 2 m/s
Message transmission time	0.1 second

Table 4.2 Mobility parameters

4.4.2 6SB as good as OFP in high densities

Figure 4.4 and 4.5 present respectively the delivery and forward ratios of the *CBS*, *OFP*, *6SB* and *PAMPA* algorithms in a static setting *without mobility.*

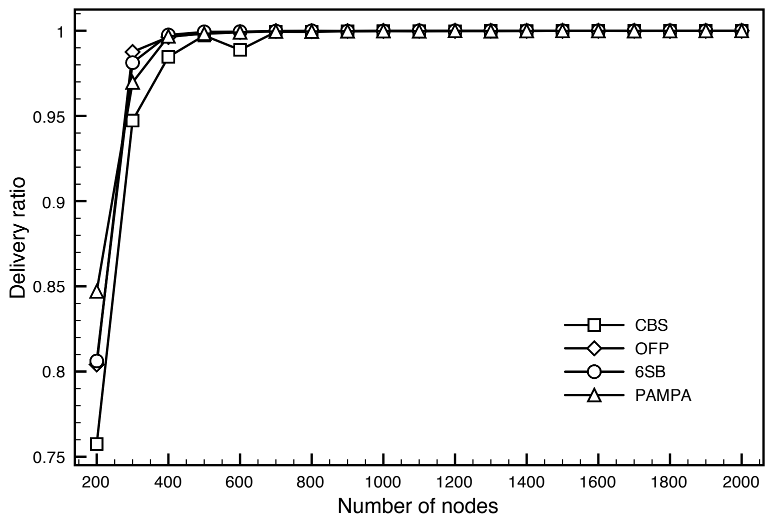


Fig. 4.4 Delivery ratio in high densities without mobility

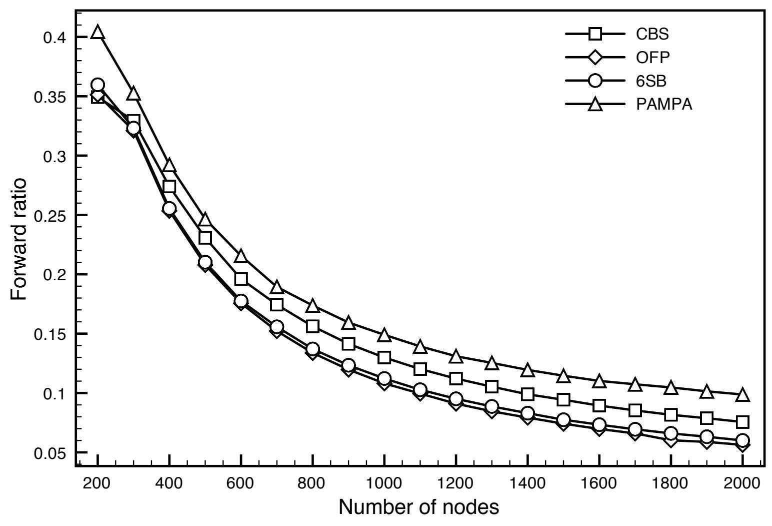


Fig. 4.5 Forward ratio in high densities without mobility

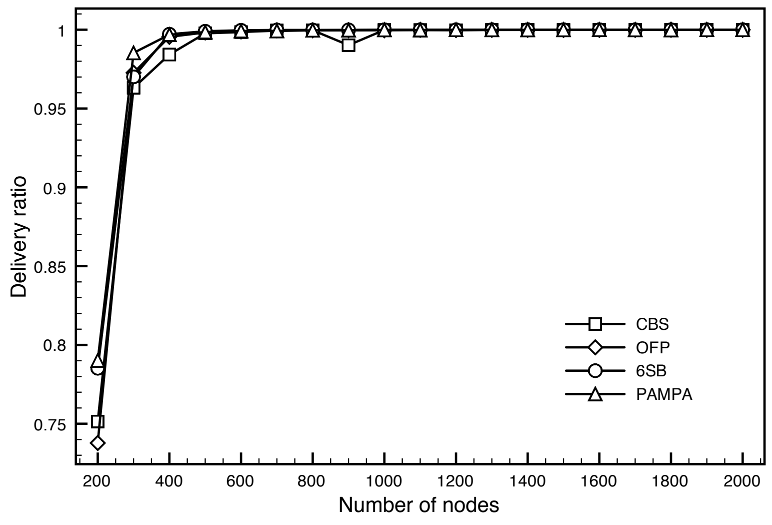


Fig. 4.6 Delivery ratio in high densities with mobility

These results show that all evaluated algorithms perform well, between 5% to 40% of the nodes have to forward the messages. In high densities (more than 600 nodes), the different algorithms have no influence on the delivery ratio that is equal to 1, because there is enough nodes to propagate the message to the whole networks. The only difference is the forward ratio. The *OFP* algorithm is the one with the best forward ratio, closely followed by *6SB*. These two algorithms are clearly the best

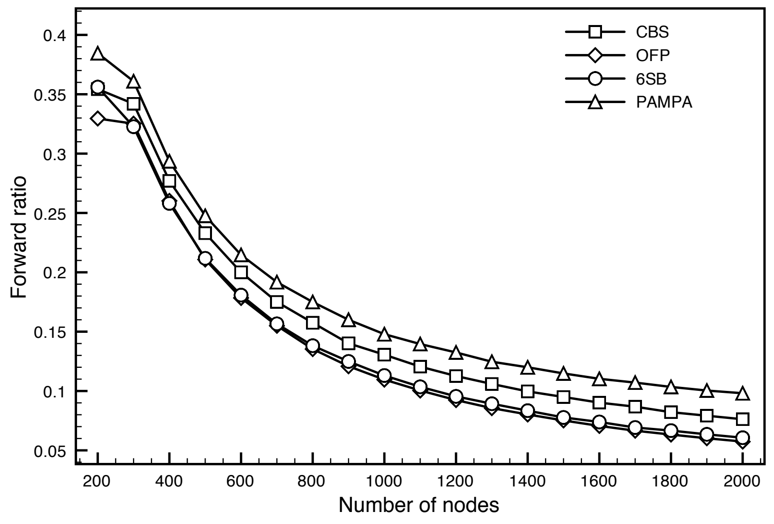


Fig. 4.7 Forward ratio in high densities with mobility

in terms of efficiency. *CBS*, despite its simplicity, turns out to be also very competitive. The *PAMPA* algorithm, which is more complicated, demonstrates poorer performances. Figure 4.6 and 4.7 present the same results in a mobile context. Mobility has almost no influence on the results. This is due to the significant difference between the slow walking speed at which nodes move and the high message propagation speed.

These results convey the fact that in high densities, when nodes have around 18 or more neighbors (above 600 nodes in our simulation settings), all evaluated algorithms are reliable and *6SB* performs as well as *OFP*, which is the best algorithm in terms of efficiency. Mobility has no influences on the results.

4.4.3 6SB better than OFP in low densities

Figure 4.8 and 4.9 present respectively the delivery and forward ratios of *CBS*, *OFP*, *6SB* and *PAMPA* in low densities (until 350 nodes). *PAMPA* and *6SB* have the highest delivery ratios. *CBS* has a delivery ratio, which is always the lowest, but it tend to join *PAMPA* and *6SB* in higher densities. *OFP* has an intermediate delivery ratio. It begins at the same low value than *CBS*, but becomes reliable more rapidly. In terms of efficiency, *OFP* has always the lowest forward ratio. *CBS* has a forward ratio a little bit higher, followed then by *6SB* and *PAMPA* in the last position. When the density increases, these ratios tend to change; *6SB* became more efficient than *CBS*.

These results show clearly that there is a tradeoff between reliability and efficiency in lower density settings. *PAMPA* and *6SB* focus on reliability, since they have a higher delivery ratio for a higher forward ratio. *OFP* and *CBS* focus on efficiency, with a lower forwarding ratio but also lower a delivery ratio.

In order to measure this tradeoff, we have computed the number of delivers gained with one more forward in comparison to OFP, which is the best algorithm in term of forward ratio (see Figure 4.10). This is the *utility* of having one more forwarder. *CBS* is the worst algorithm because it has the lowest gain for the higher variability of the results. *PAMPA* has a small utility just over 0. *6SB* has the best utility with values above 1. This means that for one more forward, *6SB* will generate more than one extra deliver. So *6SB* allows to have a better delivery ratio than *OFP* for the lowest overhead in term of forward. *6SB* offers the best tradeoff between reliability and efficiency in lower densities.

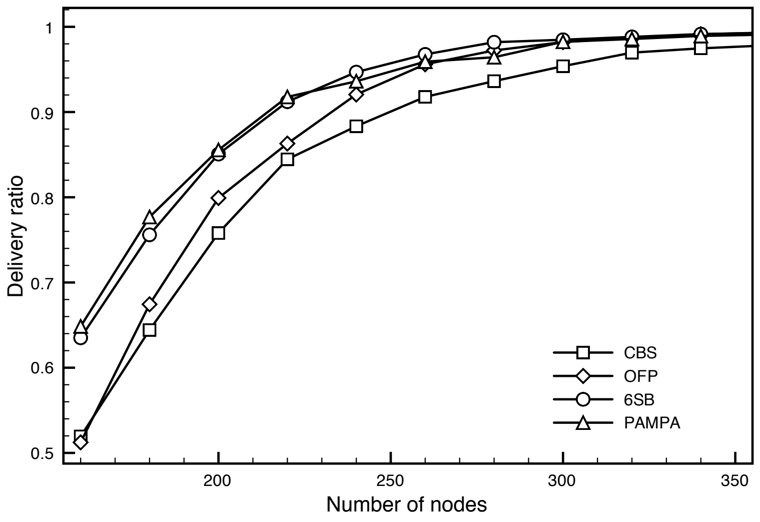


Fig. 4.8 Delivery ratio in low densities

4.5 Concluding remarks

Finding the right algorithm for broadcasting in a MANET is not a trivial task. The problem of broadcasting in MANETs is a tradeoff between reliability and efficiency. Reliability is measured by the number of nodes that deliver the message and efficiency by the number of nodes that forward the message. A maximum of nodes have to deliver the message, while we try to minimize the number of nodes that forward

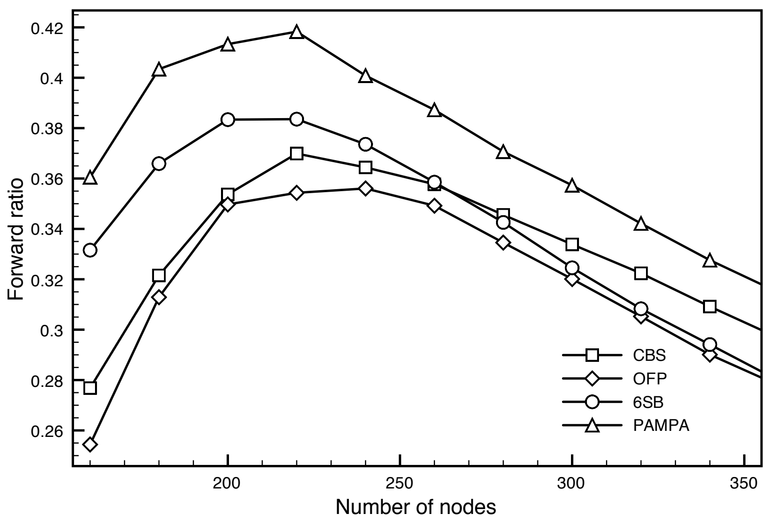


Fig. 4.9 Forward ratio in low densities

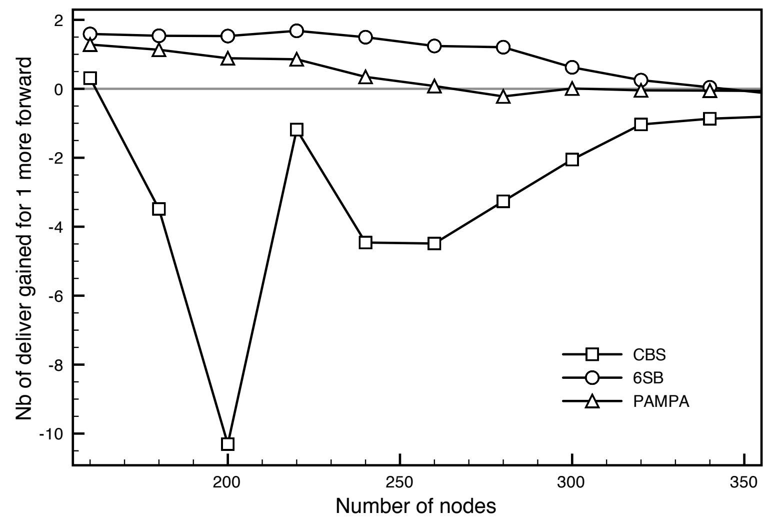


Fig. 4.10 Number of deliver gained for one more forward in comparison to OFP

this message. By optimizing the number of forwarders, we save resources, such as battery power, but we increase the risk that some nodes do not receive the message.

The main contribution of this paper is *six-shot broadcast* (*6SB*), a context-aware communication algorithm using location information to fine-tune its forwarding process. Our extensive performance evaluations show that *6SB* competes with the most efficient algorithms in high densities and offers increased reliability at a reasonable price in low densities.

We intend to extend this research in two directions, first we will have a look at a practical implementation of the *6SB* algorithm and its use on mobile devices for a concrete mobile application. Second, we will further examine *6SB* theoretically by conducting simulations in different contexts and by integrating the *6SB* algorithm into other broadcast-based algorithms, such as *FIFO-broadcast*, or *atomic-commit.*

References

[1] Area-based beaconless reliable broadcasting in sensor networks. *IJSNet*, 1(1/2):20–33, 2006.

[2] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[3] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM.

[4] Qing Huang, Yong Bai, and Lan Chen. Efficient lightweight broadcasting protocols for multi-hop ad hoc networks. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium*, pages 1 – 5, 2006.

[5] Hugo Miranda, Simone Leggio, Luis Rodrigues, and Kimmo Raatikainen. A power-aware broadcasting algorithm. In *Proceedings of the 17th annual IEEE international symposium on personal, indoor and mobile radio communications (PIMRC 06)*, pages 1–5, 2006.

[6] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping 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, New York, NY, USA, 1999. ACM.

[7] Distributed Computing Group of ETH Zurich. Sinalgo: a simulation framework for manets. *http://dcg.ethz.ch/projects/sinalgo/*.

[8] Vamsi Paruchuri, Arjan Durresi, and Raj Jain. Optimized flooding protocol for ad hoc networks. *CoRR*, cs.NI/0311013, 2003. informal publication.

Chapter 5
Six-shot Multicast: A Location-aware Strategy for Efficient Message Routing in MANETs

Published in the proceedings of the 9th IEEE International Symposium on Network Computing and Applications (NCA)
Cambridge, MA, USA, July 15-17, 2010
Publisher: IEEE Computer Society
ISBN: 978-0-7695-4118-1
Received the best student paper award.

Abstract In this paper, we introduce *six-shot multicast* (6Shot), a location-aware multicast algorithm devised for mobile ad hoc networks. Multicast is a *one-to-many* communication scheme and has largely been studied in such networks. Indeed, this communication primitive can be used as a building block for popular services, such as data streaming or group communication. The particularity of 6Shot is the location-aware routing scheme of its implementation, which offers improved efficiency in terms of message overhead compared to existing algorithms, for a reasonable cost in terms of reliability.

5.1 Introduction

Message transmission in the dynamic environment of a mobile ad hoc network (MANET) is a crucial and challenging endeavour and has therefore caught the attention of the research community for quite some time. Three main types of message transmission protocols exist. First, *broadcast* protocols, or one-to-all message diffusion, where a message is sent to all nodes in the network. Second, *unicast* protocols, or one-to-one message diffusion, where a message is only sent to one node in the network. Third, *multicast* protocols, or one-to-many message diffusion, where a message is sent to a subset of nodes in the network generally called the *multicast group*.

In MANETs, mobile nodes solely communicate with each other wirelessly. Within a node's wireless transmission range, message diffusion is trivial since all emitted messages are received by all nodes. The endeavor becomes challenging when a node wants to communicate with nodes outside its transmission range. In that case, one or several nodes will have to retransmit the message in what is called a *multi-hop* communication. In MANETs multi-hop communication is considered to be the rule rather than the exception, therefore message diffusion protocols must be devised accordingly. Furthermore, the mobility of nodes as well as the scarcity of their resources in terms of bandwidth, memory and processing power adds to the challenge in encouraging communication protocols to minimize the message overhead they generate.

This paper presents a novel location-aware multicast algorithm called *six-shot multicast* (6Shot). The multicast problem in the context of a MANET can be solved by two naive approaches: broadcast and multi-unicast. The broadcast approach consists in broadcasting a message m containing a *multicast group name* to all nodes in the network using one of the existing broadcasting algorithms [1, 3, 4, 5, 10, 12, 14]. Multi-unicast consists in sending multiple unicast message – one to *each* multicast group member – using an existing unicast algorithm, such as AODV [15] or DSR [6]. Each of these approaches is efficient in terms of message overhead in an extreme case; broadcasting being efficient when all nodes in the network are part of the multicast group and multi-unicast being efficient in the case where there is only one group members. 6Shot aims at providing an efficient implementation for a broad range of multicast group members. More precisely, it aims at offering a lower message overhead than existing multicasting algorithms for similar reliability no matter the number of multicast group members and similar efficiency to unicast and broadcast algorithms in the extreme cases where either every node is part of the group or only one is part of it.

In order to achieve high efficiency, 6Shot uses location information in a unique way in its routing process. Location-awareness is becoming more and more popular with the advent of mainstream location-aware mobile devices. While location-aware applications have emerged to enhance user experience, some communication protocols have become location-aware to facilitate the task of developers and others, like 6Shot, in order to increase their efficiency.

In the remainder of this paper, Section 5.2 presents the 6Shot Algorithm, while Section 5.3 presents a classification and comparison of our algorithm with related work. Section 5.4 then extensively discusses the performance of 6Shot, comparing it with other existing algorithms. Finally, Section 5.5 concludes the paper and offers an outlook on future research directions.

5.2 Algorithmic description

The 6Shot Algorithm offers the primitives listed hereafter and is built on top of two underlying services: a *location service*, used in the routing process, and a *medium access control layer*. Figure 5.1 depicts this layered architecture.

- JOIN($P(m)$): joins a group specified by predicate $P(m)$.

- LEAVE($P(m)$): leaves a previously joined group.
- MULTICAST(m): multicasts a message m.
- DELIVER(m): called back when message m is received and matches any predicate passed to JOIN.

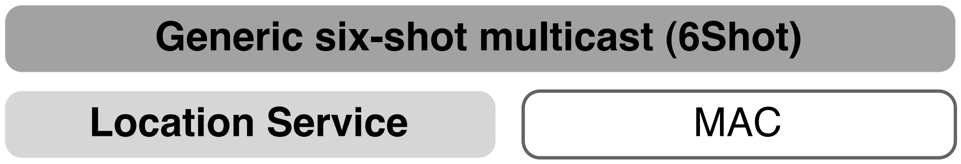


Fig. 5.1 6Shot layered architecture

Location service. This service allows access to the node's geographical location via a location sensor such as a GPS or location beacons. The location service offers the following two primitives:

- LS-GETPOSITION(): returns the host's current location;
- LS-GETDISTANCE(l_1, l_2): returns the distance from l_1 to l_2.

MAC layer. In MANETs, the *MAC* layer is used to broadcast and receive messages in a node's *neighborhood*, the neighborhood being defined by the transmission range dependent on the wireless technology being used and by the physical environment. We model this communication capability via the two following primitives:

- MAC-BROADCAST(m): broadcasts message m to all nodes located in the sender's neighborhood;
- MAC-DELIVER(m): called back when m is received.

5.2.1 Algorithmic overview

Like many multicast algorithms, 6Shot builds and maintains an overlay used to route messages. Its uniqueness resides in the fact that it is built as hybrid between a DAG-based unicast protocol, such as CCBR [1] and TORA [13], and a location-aware broadcast protocol, such as OFP [14] or 6SB [3]. Hereafter, we present the implementation of our service breaking it down into three key processes, namely *overlay creation*, *message routing* and *overlay maintenance*. These processes are detailed in Algorithms 5.9 to 5.12, alongside concrete examples depicted in Figures 5.2 to 5.4.

5.2.2 Overlay creation

Traditional multicast algorithms generally build one overlay per multicast group. As we want to implement a generic service with flexible group membership this approach is not suitable. 6Shot creates a *directed acyclic graph*, or DAG, for each host when it calls JOIN. A DAG for a given host – called the *sink* – consists in evaluating, for every host in the network, the distance in number of hops – called *height* – that separates it from the sink. When the DAG is built, all heights form a gradient converging towards the sink. Messages aimed at the sink flow down the gradient from a higher host to a lower host until they reach the destination at the bottom of the gradient. Overlay information is stored in each host's routing table RT, which keeps track of routes to different destinations. We represent a (partial) route $r = \langle h, g \rangle$ abstractly as a destination h and its proximity in terms of network hops or its height, denoted g. It should be noted that a route is really just a position on the path to the sink.

The algorithm. As detailed in Algorithm 5.9, this scheme translates into broadcasting join messages through the network with a hop counter to create routing table entries. This process only uses the underlying MAC layer. When JOIN is called with the predicate $P(m)$ as parameter, a new message ID is computed and added to the list of received join messages. Then a message containing the message ID ($msgID$), the predicate ($P(m)$) and the tuple representing route to the host ($\langle myID, 0 \rangle$) is broadcasted over the MAC layer (lines 7-10). When such a message is delivered for the first time through the MAC-DELIVER callback, the tuple containing the message ID, the predicate and the host are added to the list of received join messages (lines 11-13). Then the height contained in the message is incremented and a new entry is created in the routing table (lines 14-15).

Then, rather than directly forwarding the message in the same way a simple flooding algorithm would do, the process gets into a random waiting phase (line 16). During this waiting phase, the host listens for possible retransmissions of the same message. If such a retransmission occurs, the boolean entry for the message in the *shouldForward* list is set to false and the message will never be forwarded (line 20). If the boolean entry is still true after the waiting time, the message is forwarded (lines 17-18). This broadcasting technique is similar to the one proposed in the counter-based broadcasting scheme [10] which has been shown to dramatically decrease the message load of broadcasting algorithms with a reasonable cost in terms of reliability.

An example. Figure 5.2 illustrates the creation of a DAG for two group members in a MANET containing eight nodes. Figure 5.2.1 depicts the network topology. In Figure 5.2.2, Bruno calls JOIN, which initiates a network-wide broadcast and all nodes receiving the message add an entry in their routing table containing the ID of the source, namely Bruno, and its proximity in number of hops, e.g., Karine is

```
1: uses MAC

2: Init:
3:     $RT \leftarrow \emptyset$                                        {routing table}
4:     $RT \leftarrow RT \cup \{\langle myID, 0\rangle\}$                {adds myID as sink}
5:     $joinMsgs \leftarrow \emptyset$                                  {list of received msgs}
6:     $shouldForward \leftarrow \langle true, true, ...\rangle$         {indexed by msg}

7: To execute JOIN($P(m)$) :
8:     $msgID \leftarrow$ GETNEWMSGID()
9:     $joinMsgs \leftarrow joinMsgs \cup msgID$
10:    MAC-BROADCAST($\langle msgID, P(m), myID, 0\rangle$)

11: upon MAC-DELIVER($\langle msgID, P(m), h, g\rangle$) do
12:    if $\neg\exists\langle msgID', P(m)', h'\rangle \in joinMsgs \mid msgID' = msgID$ then
13:        $joinMsgs \leftarrow joinMsgs \cup \{\langle msgID, P(m), h\rangle\}$
14:        $g \leftarrow g + 1$                                         {increments height}
15:        $RT \leftarrow RT \cup \{\langle h, g\rangle\}$                 {adds the new route}
16:        WAIT(RANDOM)                                                 {waits for a random delay}
17:        if $shouldForward[msgID]$ then
18:            MAC-BROADCAST($\langle msgID, P(m), h, g\rangle$)
19:    else
20:        $shouldForward[msgID] \leftarrow false$
```

Algorithm 5.9 6Shot overlay creation

two hops away from Bruno therefore her table entry is B:2. Figure 5.2.3 shows what happens when Phil calls JOIN. A DAG is also created for him and every node has now an entry for both Bruno and Phil in their routing tables.

5.2.3 Message routing

The routing process in 6Shot is *blind*, which means that a sender does not know which one of its neighbors will be the next forwarder of the message, hence multicast messages are always blindly broadcast to all neighbors. These neighbors will then set a *schedule* for the forwarding of the message if they are located on the route – lower on the gradient – to at least one of the message's multicast group members. The scheduled forwarding time is not set randomly but, similarly to the 6SB broadcasting algorithm proposed in [3], proportionally to the proximity of the receiver to the closest of six geographical locations, called *targets*.[1] The closer the neighbor is to the target, the sooner is the scheduled forwarding time. These six targets are the

[1] The six targets form a hexagon around the sender of the message. The hexagonal shape allows to minimize the number of circles needed to cover a given geographical range.

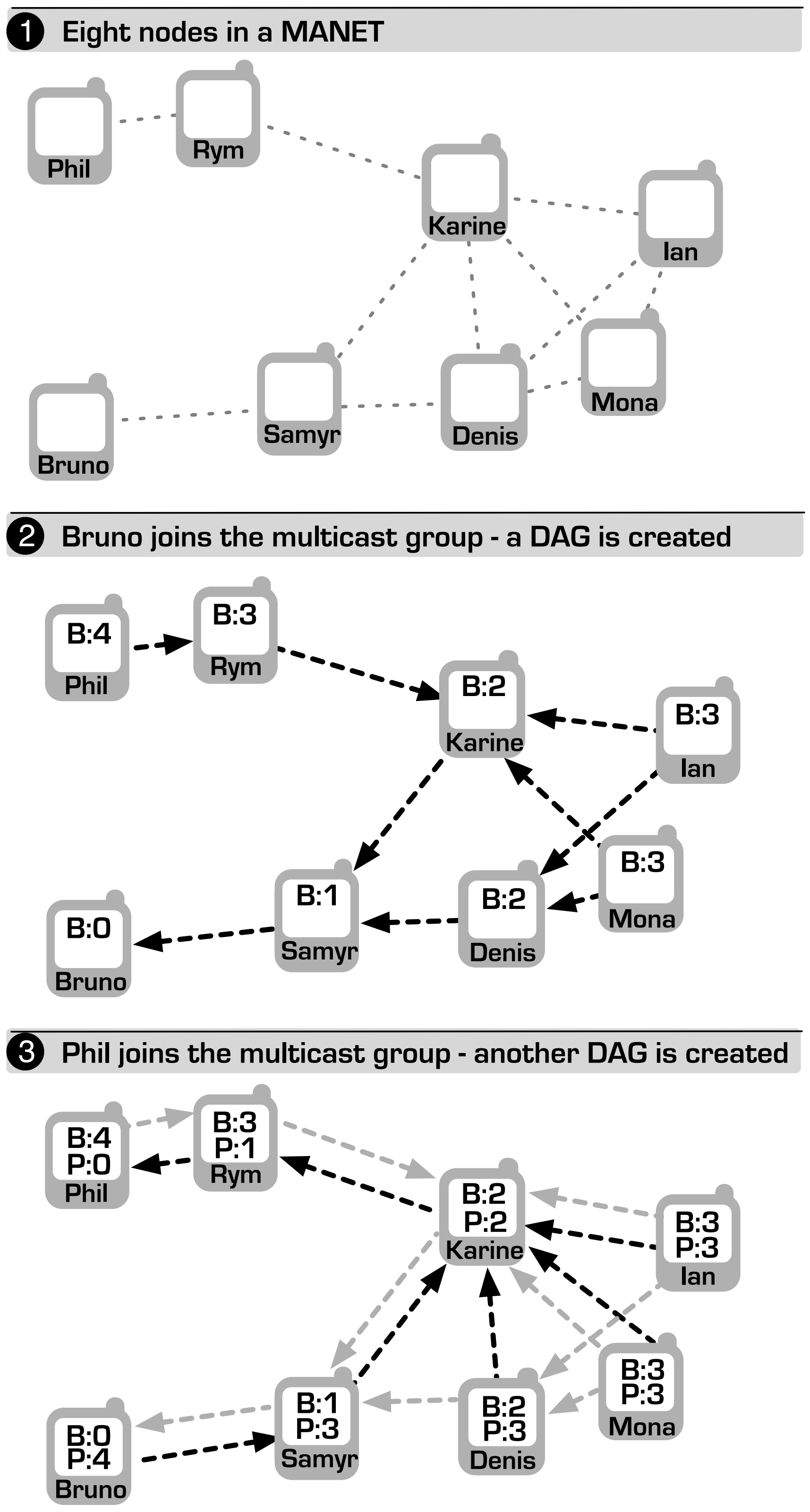


Fig. 5.2 6Shot overlay creation example

summits of a hexagon centered around the sender, the size of its transmission range as illustrated in Figure 6.1. Similarly to the broadcasting technique used to create the overlay, on the scheduled forwarding time, m is only forwarded if no other retransmission of m was received during the wait.

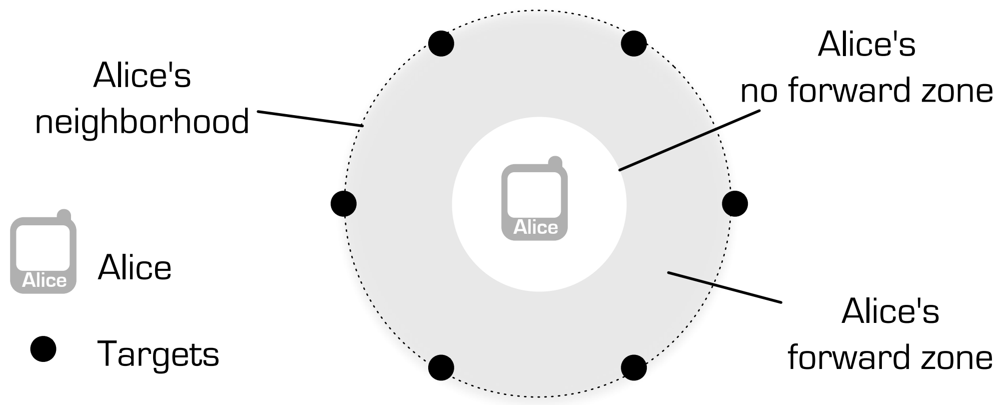


Fig. 5.3 Targets

The algorithm. Algorithm 5.10 details this process. Upon initialization, the *schedules* table is created. An entry in this table contains a message m and a time t representing the time at which m is to be forwarded. In order to multicast a message m, MULTICAST is called with m as parameter. The first and central operation is the matching between received predicates and m to compute group membership. In order to do so, m is evaluated by all received predicates gathered in the *joinMsgs* list. For every predicate that holds for m, the associated host h is considered as a group member and it is added to the message's routes with its height g. Finally, before m is broadcast on the MAC layer, the location of the sender is added to m via the LS-GETPOSITION primitive (Algorithm 5.10 lines 4-9).

When m is received, DELIVER is triggered, m is delivered if the host is part of the multicast group and a schedule for the forwarding of m is created if the message was received for the first time and if the host is on the route to at least one group member. Otherwise, if m is a retransmission of a previously received message, its forward time is set to -1 to indicate that it should never be forwarded (Algorithm 5.10 lines 10-27). The scheduled forwarding time is computed based on the proximity of the receiver to the closest of the six geographical targets, i.e., the six summits of the hexagon centered around the sender, the size of the transmission range (Algorithm 5.11 lines 1-14). The DELAY function returns a delay proportional to the distance *dist* passed as parameter, the shorter the distance, the shorter the delay.

```
1: uses LS, MAC

2: Init:
3:    $schedules \leftarrow \emptyset$                          {scheduled forward time by message}

4: To execute MULTICAST($m$) :
5:    for all $\langle msgID, P(\ ), h\rangle \in joinMsgs \mid P(m)$ do
6:       if $\exists\langle h', g\rangle \in RT \mid h' = h$ then
7:          $m.routes \leftarrow m.routes \cup \{\langle h, g\rangle\}$
8:    $m.senderLoc \leftarrow$ LS-GETPOSITION()
9:    MAC-BROADCAST($m$)                                         {forwards msg}

10: upon MAC-DELIVER($m$) do
11:    if $\langle h, g\rangle \in m.routes \mid g = 0$ then                      {is sink}
12:       call DELIVER($m$)                                      {triggers callback}
13:       $m.routes \leftarrow m.routes \setminus \{\langle h, g\rangle\}$                  {updates routes}
14:    SETSCHEDULE($m$)                                           {forwards}

15: procedure SETSCHEDULE($m$)
16:    if ISONROUTE($routes$) then
17:       if $\exists\langle m', t\rangle \in schedules \mid m' = m$ then
18:          $schedules \leftarrow schedules \setminus \{\langle m', t\rangle\}$
19:          $schedules \leftarrow schedules \cup \{\langle m', -1\rangle\}$
20:       else
21:          $t \leftarrow$ GETCURRENTTIME + GETDELAY($m$)
22:          $schedules \leftarrow schedules \cup \{\langle m', t\rangle\}$

23: function ISONROUTE($routes$) :
24:    for all $\langle h, g\rangle \in routes$ do
25:       if $\exists\langle h, g'\rangle \in RT \mid g' < g$ then
26:          return true
27:    return false
```

Algorithm 5.10 6Shot algorithm – part 1

In order to evaluate which messages need to be forwarded at a certain moment in time, the PROCESSSCHEDULE task is executed periodically. It goes through the list of schedules and updates the routes for every ripe message and adds the location of the host before forwarding it (Algorithm 5.11 lines 15-27). The UPDATEROUTES function is central to the algorithm in that it readjusts the message's heights after every forward in order for the message to continue to flow downwards toward at least one of the group members.

An example. Figure 5.4 shows an example of 6Shot in use. Mona uses the multicast primitive to send a message m to Bruno and Phil. First Mona broadcasts m to her neighbors (Figure 5.4.1). Ian discards m since he is located upstream on the gradient

```
1: function getDelay($m$) :
2:     $myLoc \leftarrow$ ls-getPosition()                                    {gets my location}
3:     $d \leftarrow \perp$                                                  {initiates distance d}
4:     for all $target \in$ getTargets($m.senderLoc$) do
5:         if (ls-getDistance($myLoc$, $target$) $< d$) $\vee$ ($d = \perp$) then        {if target is closer}
6:             $d \leftarrow$ ls-getDistance($myLoc, target$)
7:     return delay($dist$)

8: function getTargets($senderLoc$) :
9:     $targets = \emptyset$
10:    for $0 <= i < 6$ do                                                   {creates six targets}
11:        $x_i \leftarrow senderLoc.x + sin(30 + 60 * i) * range$
12:        $y_i \leftarrow senderLoc.y + cos(30 + 60 * i) * range$
13:        $targets \leftarrow targets \cup \{\langle x_i, y_i \rangle\}$                  {adds the target}
14:    return $targets$                                                     {returns the targets}

15: task processSchedule
16:    periodically do
17:        for all $\langle m, t \rangle \in schedules$ do
18:            if $t \neq -1 \ \wedge \ t \leq$ getCurrentTime then
19:                $m.routes \leftarrow$updateRoutes($m.routes$)
20:                $m.sender \leftarrow$ ls-getPosition
21:                mac-broadcast($m$)                                        {forwards m}

22: function updateRoutes($routes$) :
23:    for all $\langle h, g \rangle \in routes$ do
24:        if $\exists \langle h', g' \rangle \in RT \mid h' = h \wedge g' < g$ then
25:            $routes \leftarrow routes \backslash \{\langle h, g \rangle\}$
26:            $routes \leftarrow routes \cup \{\langle h', g' \rangle\}$
27:    return $routes$
```

Algorithm 5.11 6Shot algorithm – part 2

to both Bruno and Phil. Karine and Denis on the other hand are located downstream and schedule the retransmission of m proportionally to their proximity to one of the six geographical targets computed from Mona's location. As Karine is closer to a target than Denis, she will be the first to forward m (Figure 5.4.2). When Denis receives her message, he aborts the scheduled forward, since it would be redundant. When Samyr and Rym receive the message, they schedule the transmission since they are downstream each on the gradient of one destination. Samyr is the first to retransmit since he is closer to a target (Figure 5.4.3). Bruno delivers m and finally Rym forwards m and Phil delivers it as well (Figure 5.4.4).

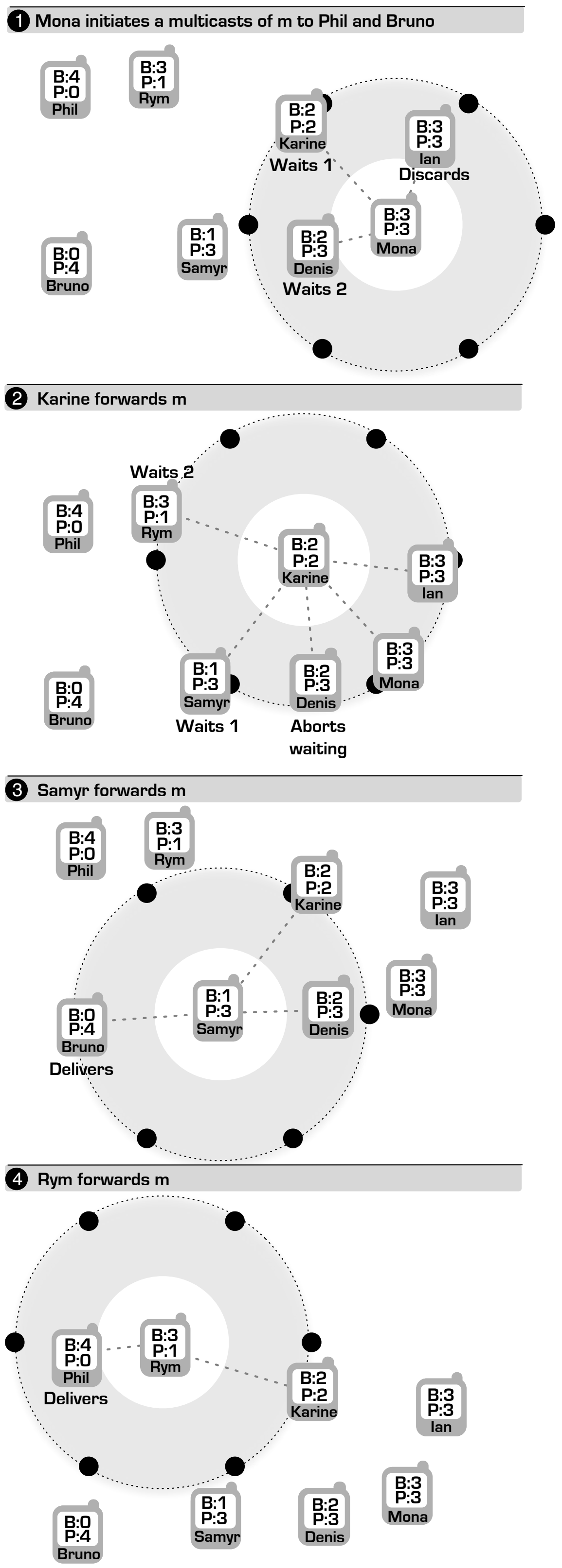


Fig. 5.4 6Shot message routing example

5.2.4 Overlay Maintenance

The DAG is maintained by updating the nodes' heights on the gradient to all sinks periodically. This process is based on the path reverse technique [2] where a node reverses its incoming links (linking a node to a higher node on the gradient) when there are no more outgoing links (linking a node to a lower node on the gradient) and readjusts its height. However, 6Shot does not create explicit links so there is no real link reversal, but rather a new height evaluation. The idea is that for a given sink, a host different from the sink gets the heights of all its neighbors and set its own height to a level just above the lowest neighbor.

```
1: uses MAC

2: Init:
3:     $neighborRTs \leftarrow \emptyset$                              {neighbors' routing tables}

4: task OverlayMaintenance
5:     repeat periodically
6:         if $neighborRTs \neq \emptyset$ then
7:             for all $\langle h, g\rangle \in RT$ do                              {for each route}
8:                 if $g \neq 0$ then                             {if the node is not the sink}
9:                     $RT \leftarrow RT \backslash \{\langle h, g\rangle\}$
10:                    $g \leftarrow \perp$
11:                    for all $RT' \in neighborRTs$ do
12:                        if $\exists\langle h', g'\rangle \in RT' \mid h' = h \wedge g' < g \vee g = \perp$ then
13:                            $g \leftarrow g'$
14:                    $RT \leftarrow RT \cup \{\langle h, g+1\rangle\}$
15:            $neighborRTs \leftarrow \emptyset$
16:        mac-broadcast($RT, localhost$)

17: upon mac-deliver($\langle someRT, h\rangle$) do
18:     if $h \neq localhost$ then
19:         $neighborRTs \leftarrow neighborRTs \cup someRT$
```

Algorithm 5.12 6Shot overlay maintenance

The algorithm. Algorithm 5.12 details the maintenance process. Periodically, every host updates its routing table and broadcasts it to all neighbors (lines 4-16). When a routing table is delivered with the reference to the sending host through the MAC-DELIVER primitive, it is added to the *neighborRTs* set containing all of the neighbors' routing tables (lines 17-19). During the updating phase, the process erases all entries in its routing table except the one with itself as sink and goes through all the received routing tables to find the lowest neighbor for each sink and re-creates an entry in its routing table for that sink with a height higher by one unit to the height of the lowest neighbor. Like the path reverse technique, it might take several updates before the DAG returns to a fully coherent state after a disruption.

About disconnections. It is assumed that all partitions eventually heal. Our maintenance mechanism does not store messages and forward them to nodes that were disconnected. In order to address this issue a simple acknowledgement mechanism can be implemented on top of our algorithm.

5.3 Related work

We compare 6Shot to different existing multicast algorithms [1, 7, 8, 9, 16, 17, 18] and point out the differences along four dimensions: (1) the topology of the overlay, (2) the strategy used to create the overlay, (3) the type of overlay maintenance used, and (4) the level of route-awareness. Table 5.1 depicts these dimensions and the classification of the different algorithms.

5.3.1 Overlay topology

All presented multicast algorithms share one common characteristic in that they build some sort of overlay network used to route multicast messages to multicast group members. The goal of this overlay is to build some optimal path in order to easily reach all group members. As depicted in Table 5.1, most systems build either a tree (AMRoute [18], AMRIS [17], LAM [8], MAODV [16]) or a mesh (ODMRP [7], CAMP [9]), encompassing all group members. Links between two members of a mesh or of a tree are typically computed using a shortest path algorithm. In all of the tree-based algorithms, the root of the tree is an arbitrary multicast group member. CCBR [1] and 6Shot depart from these approaches and builds a directed acyclic graph (DAG) *per destination* in a somewhat similar fashion to the DAG created in the TORA [13] unicast algorithm. Contrary to the DAG structure used in TORA, where each node keeps track of the ingoing and outgoing connections, our algorithm only monitors the height of each node on the gradient to a destination. This makes these algorithms lightweight although several DAGs are built.

5.3.2 Overlay creation strategy

Routing algorithms in general can have two strategies when it comes to creating their overlay: a proactive strategy and a reactive strategy. In the reactive strategy, also referred to as *on-demand*, the overlay is created when the first sender is about

	Topology	**Creation**	**Maintenance**	**Routing**
6Shot	dag	proactive	decentralized	blind
CCBR	dag	proactive	centralized	blind
AMRIS	tree	proactive	decentralized	sighted
AMRoute	tree	proactive	centralized	sighted
CAMP	mesh	proactive	centralized	sighted
LAM	tree	reactive	decentralized	sighted
MAODV	tree	reactive	centralized	sighted
ODMRP	mersh	reactive	centralized	sighted

Table 5.1 Multicast algorithm classification

to multicast a message. This avoids building an overlay if no-one uses it, but it implies a certain latency when the first message is sent. Reactive algorithms include algorithms such as MAODV, ODMRP and LAM. In opposition, in the proactive strategy, the overlay is built as soon as a node joins the multicast group and is ready to use when a node decides to multicast a message. 6Shot is a proactive algorithm alike CCBR, AMRoute, AMRIS and CAMP.

5.3.3 Overlay maintenance

Overlay maintenance is a key issue in multicast algorithms and two main techniques stand out. First some algorithms use a centralized approach where a central node such as the tree core, or receivers in a mesh flood the network periodically to update routes, e.g., AMRoute, CAMP, MAODV, CCBR. Other algorithms, such as 6Shot, AMRIS, and LAM use a decentralized approach where nodes update routes locally. Scalability is thus improved. Local route repair algorithms offer higher scalability and are less vulnerable to the issue of single point of failure. The link reevaluation technique used in 6Shot is inspired by the path reverse technique used in TORA. In comparison with TORA's technique, which keeps track of all incoming and outgoing links and reverses incoming links when all outgoing links are broken, links in 6Shot are not monitored and heights are entirely reevaluated periodically.

5.3.4 Route-awareness

When multicasting a message in a MANET, a sender is usually aware of a part of the route a message must follows even if it only knows who the next forwarder ought to be. AMRIS, AMRoute, CAMP, LAM, MAODV and ODMRP follow this approach, which we call *sighted*. CCBR and 6Shot departs from this approach and builds upon one used by broadcasting algorithms such as the counter-based broadcast scheme (CBS) [10], where a sender does not know which one of its neighbors will forward the message. We call this approach *blind*. In this approach a sender broadcasts a message to all of its neighbors which will independently decide wether or not the message should be forwarded. Our results show that the blind approach allows to increase efficiency in terms of message load.

5.4 Performance analysis

We evaluate the performance of 6Shot by comparing it to three different algorithms. In order to evaluate its performance in very small multicast groups, we compare it to the popular AODV [15] unicast algorithm used in a multi-unicast fashion, i.e., where messages are unicast sequentially to every group member. For very large multicast groups, we then compare it to the 6SB [3] probabilistic broadcast algorithm. Furthermore, in order to evaluate its performance in between these extremes, we compare 6Shot to AMRoute[18], a multicast algorithm presented in the previous section and CCBR[1] the multicast algorithm closest to 6Shot without location-awareness.

Our results show that 6Shot performs extremely well in terms of message load from small to very large levels of multicast group members, for a reasonable cost in terms of reliability. Nevertheless, for very small levels of group members, 6Shot is outperformed by AODV and AMRoute.

5.4.1 Simulations parameters

For our performance evaluations, we choose a scenario of an event such as a conference, where every participant has a mobile device with wireless capabilities such as 802.11 WiFi and a localization system. For the simulations, we used the *Sinalgo* framework [11] written in Java. The *delivery ratio* is defined as the number of members who received the multicast message divided by the total number of members. The *message load* is the number of messages sent through the MAC layer (unicast and/or broadcast) in a simulation. The *membership ratio* is the number of multicast group members divided by the number of nodes in the network. The *delivery ratio* and *message load* are presented in function of the *membership ratio*. For the simulations, we used the parameters described in Table 5.2. For simulations with mobility, we add the mobility parameters of Table 5.3.

General parameters. Among the general parameters (Table 5.2), we use a 200 meter-wide torus on which nodes evolve. The torus shape allows to avoid bound effects and some well known problems with the Random Waypoint mobil-

Name	Value
Field size	$200m \times 200m$ (torus)
Communication range	Wifi – $20m$
Number of nodes	$200, 400$
Multicast group members	1 - every nodes
Max. waiting delay	$2s$
Message transm. speed	$0.1s$
Nodes distribution	Random
Number of simulations	100

Table 5.2 Simulation general parameters

Name	Value
Mobility model	Random waypoint (torus)
Speed	Walking speed $(1 - 2m/s)$
Pause	$0 - 10s$
Overlay Maintenance	every $0.1s$
Multicast send	$20s$ after DAG creation

Table 5.3 Mobility parameters

ity model [19]. The field is populated by 200 or 400 nodes. With a nodes density of 200, partitions may exist but a giant component is always present. With a nodes density of 400 and above, the network is fully connected and thus the results obtained do not vary much. Each node has communication capabilities with a *communication range* of 20 meters.[2] The multicast group members vary between 10% and 100% or between 2 (one sender and one receiver) and 50 members in low membership simulations. The transmission and processing time of a message is 0.1 second. For algorithms with a *waiting delay* before the retransmission of a message, the maximum waiting delay is 2 seconds. We simulate all algorithms on the same 100 different node distributions. The results shown in the following figures are the means of these 100 simulations.

Mobility parameters. In order to simulate *mobility* we use the widespread Random Waypoint (RWP) mobility model adapted to our torus-shaped map with nodes moving at walking speed (1-2m/s). When reaching their waypoint, nodes wait for a random delay between 0 and 10 seconds before fixing the next waypoint. With mobility appears the problem of overlay maintenance. The overlay maintenance is done by a repeated task (Section 5.2.4). In our simulations, this task is repeated every 0.5 seconds. Finally, a 20 seconds waiting time is set between the beginning

[2] This range might seem small with respect to the theoretical range of the WiFi technology and results from experimental tests in a building, and takes interferences into account.

of the simulation and the message multicast in order for the overlay to be created and the maintenance process tested (see Table 5.3).

5.4.2 Results without mobility

Here we describe the performance results in a setting without mobility. Figures 5.5 to 5.8 present the results of the algorithms in function of the multicast group membership ratio for densities of respectively 200 and 400 nodes.

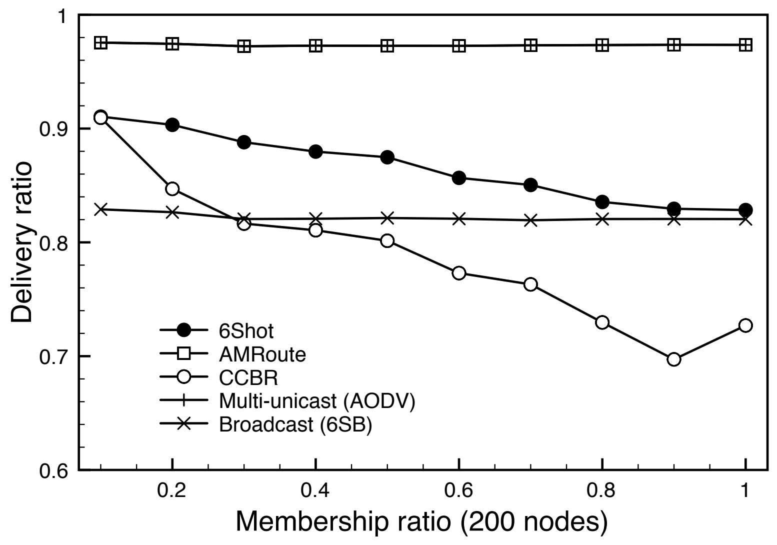


Fig. 5.5 Delivery ratio – no mobility.

Delivery ratio. AMRoute and the multi-unicast are reliable algorithms, contrary to 6Shot, CCBR and 6SB, which are probabilistic and thus not reliable by definition. Although the delivery ratio of the reliable algorithms should be equal to 1, in Figure 5.5 we observe that the delivery ratio of these algorithms are the best and almost constant to 0.97. This is due to the fact that with such a low node density there are a few links between the node and so increased probability that there are unreachable nodes when the multicast group increases, due to network partitions. 6Shot has the next best delivery ratio. Its ratio decreases with the raise of the membership ratio. This is due to the fact that there is more probability of not reaching a member when they are more members. CCBR has a delivery ratio like 6Shot with a small multicast group, but the delivery ratio quickly decrease with the increase of the membership ratio. With a membership ratio higher to 0.3, using CCBR is worse than using 6SB. The delivery ratio of 6SB is almost equal to 0.83, which is not so

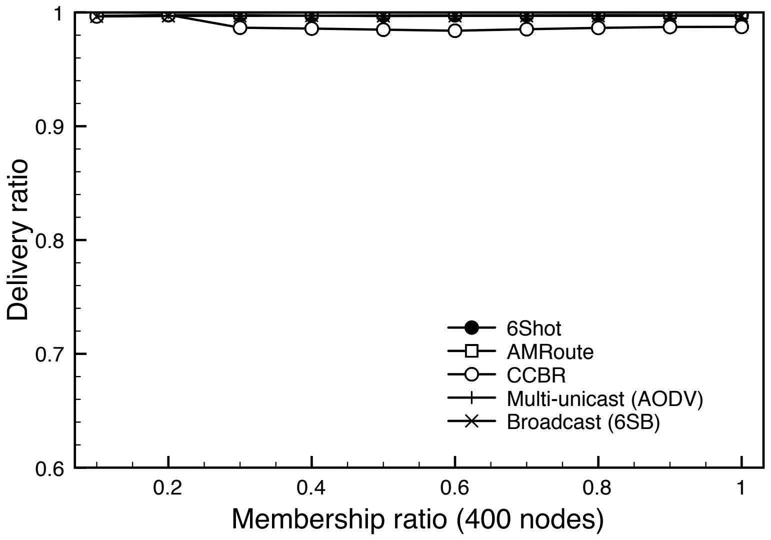


Fig. 5.6 Delivery ratio – no mobility.

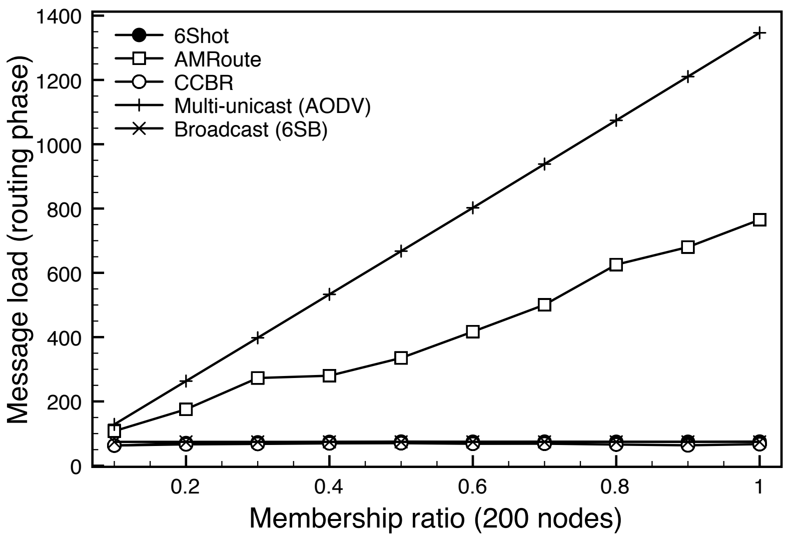


Fig. 5.7 Message load – no mobility.

bad for a probabilistic algorithm in a low densities of nodes. Low node densities, under 0.0075 nodes/m^2, create network partitions and have an impact on the delivery ratio of our 6Shot algorithm. For higher node densities, above 0.01 nodes/m^2, node density is no longer a problem and does not influence the behavior of the algorithms. Figure 5.6 demonstrates that for such densities the delivery ratio of all algorithms is almost or strictly equal to 1 except for CCBR that is near 0.98.

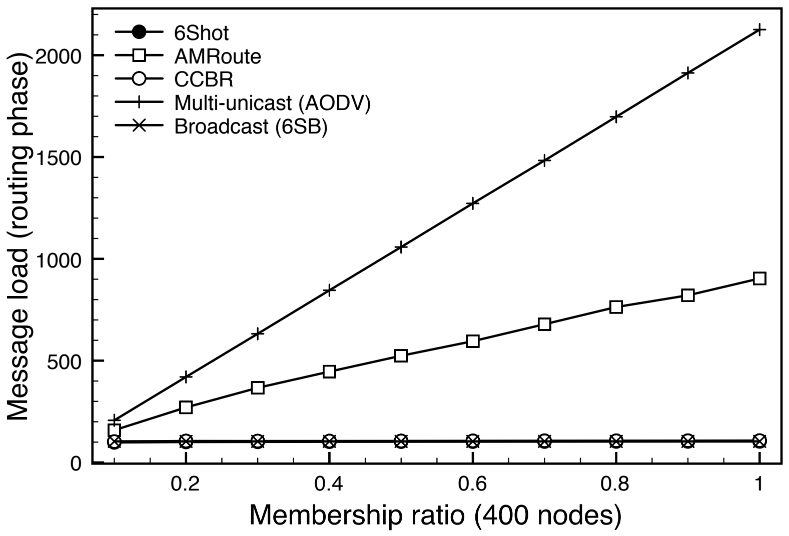


Fig. 5.8 Message load – no mobility.

Message load. Message load increases with density, but the algorithms' behavior stays the same (Figures 5.7 and 5.8). The most efficient algorithms are 6Shot, CCBR and 6SB with a constant low message load. The message loads of AMRoute and of the multi-unicast increase linearly with membership ratio, the former increasing less rapidly than the latter.

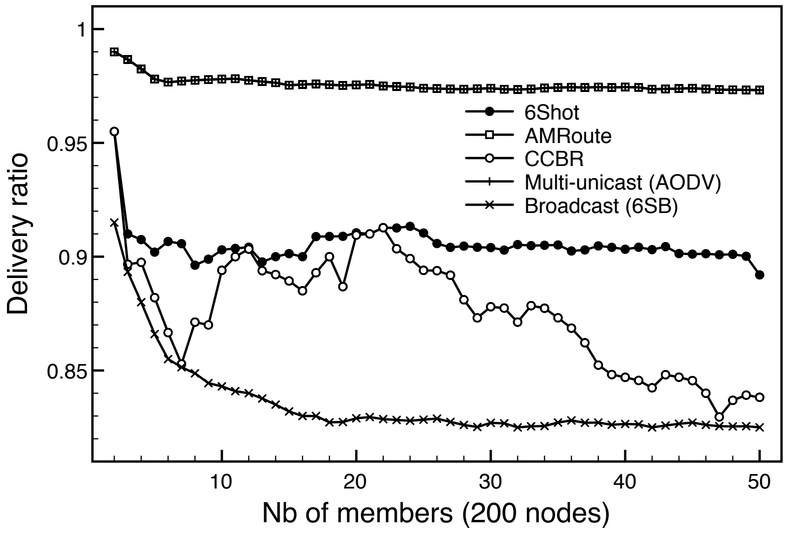


Fig. 5.9 Delivery ratio – no mobility – zoom.

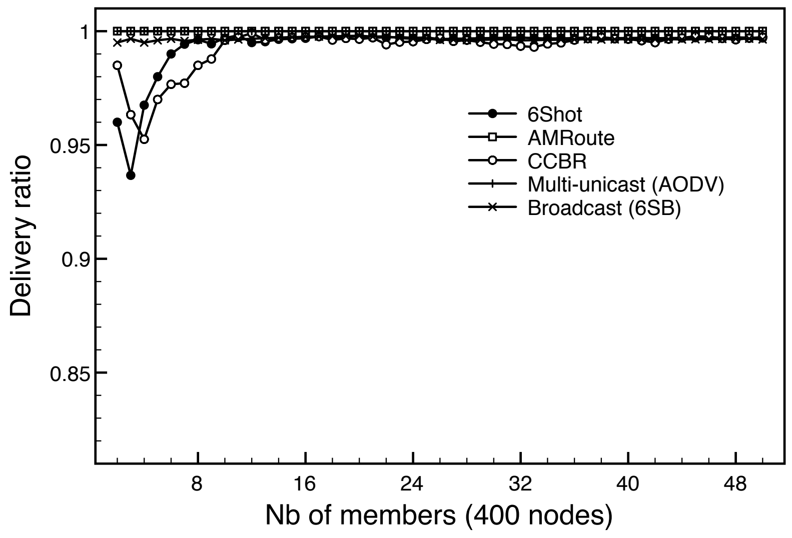


Fig. 5.10 Delivery ratio – no mobility – zoom.

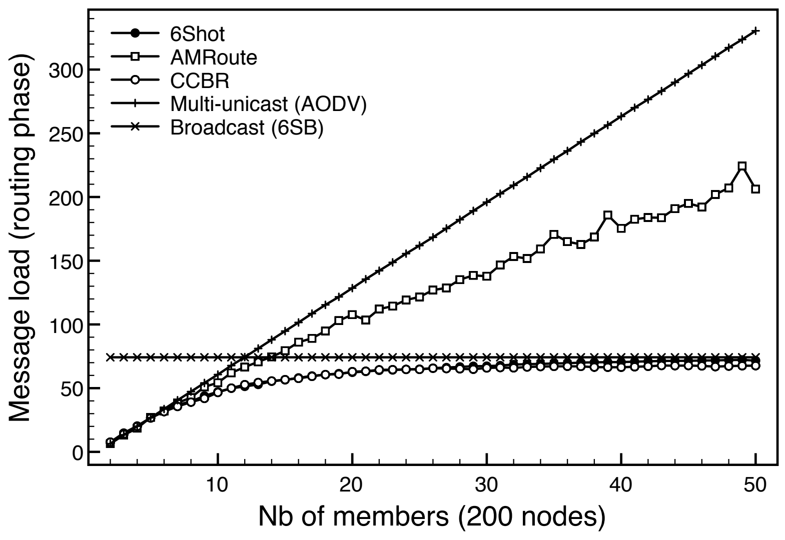


Fig. 5.11 Message load – no mobility – zoom.

Zoom on low membership ratio. Previous figures detail the delivery ratio and message load for a membership ratio between 0.1 and 1. But, as we have seen, the differences between the various algorithms are rather small where the membership ratio is low. For this reason, we zoom in Figures 5.9 to 5.12 on the previously shown

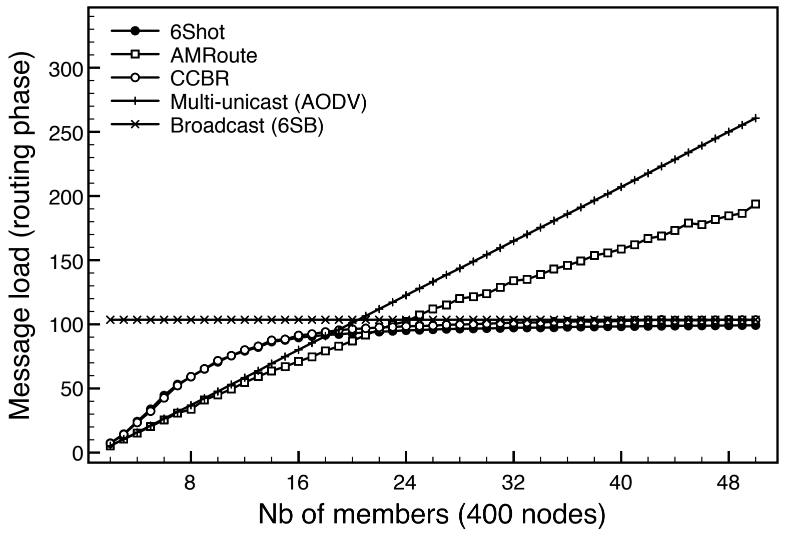


Fig. 5.12 Message load – no mobility – zoom.

results and detail the delivery ratio and message load for multicast groups between 2 (one sender and one receiver) and 50 members[3].

Regarding message load with low membership ratio, 6SB is constant. The message load of AMRoute and multi-unicast increase in a linear fashion. In contrast, the message load of 6Shot and CCBR increase in a log fashion and tend to equal the 6SB results, which implies remarkable scalability. For a density of 200 nodes (Figure 5.11), the message load of 6Shot and CCBR outperform all other algorithms for any level of group membership. Nevertheless, these results come at the cost of significantly reduced reliability of about 7% for 6Shot and over 10% for CCBR compared to AMRoute and multi-unicast. For a density of 400 nodes (Figure 5.12), the reliability of 6Shot and CCBR is up to par with the ones of AMRoute and AODV, and we can see that 6Shot and CCBR exhibit a higher overhead than these algorithms for very low membership ratios with less than 16 group members, but otherwise outperforms dramatically AMRoute and AODV.

Differences between 6Shot and CCBR.

Figure 5.13, shows the detailed results of 6Shot, CCBR and 6SB. It shows that 6Shot is preferable to both other algorithms since it offers a lower message load for the same or better reliability.

[3] These values corresponds to a membership ratio between 0.005 and 0.25 for 200 nodes and a ratio between 0.0025 and 0.125 for 400 nodes.

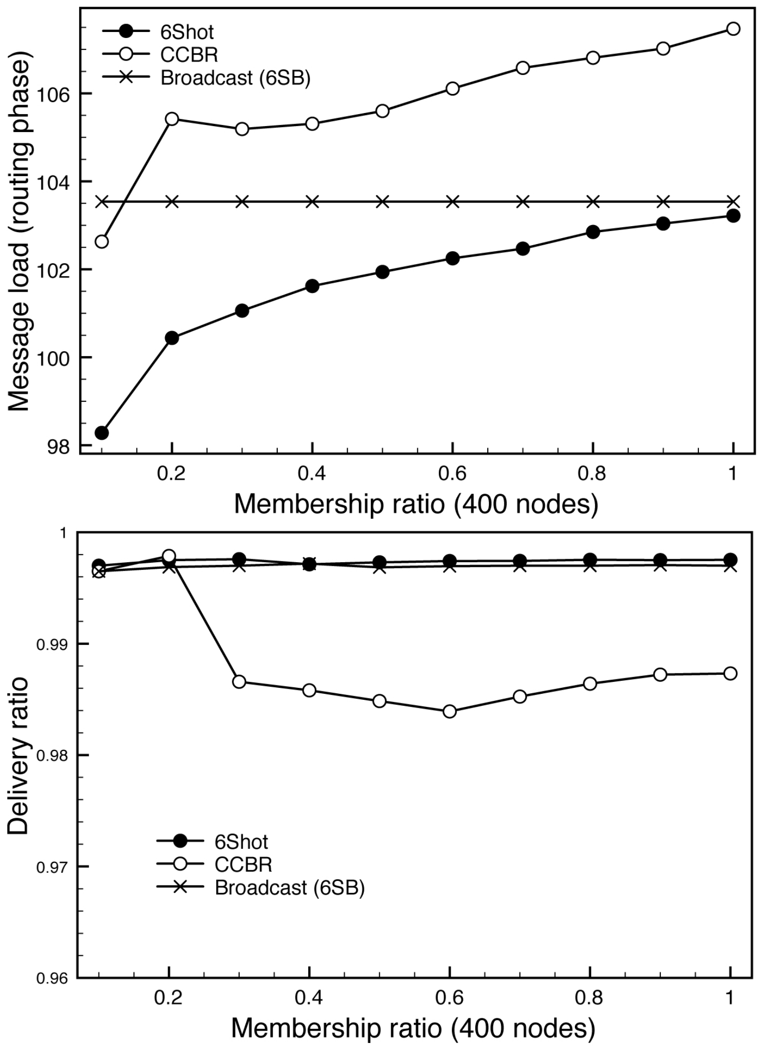


Fig. 5.13 Differences between 6Shot and CCBR – without mobility.

5.4.3 Results with mobility

The previous results analyze the algorithms in a static environment. Figures 5.14 to 5.17 exhibit the previous static results of 6Shot as well as results in a mobile environment. It shows that the delivery ratio in a low node density with a low membership ratio strongly decreases from 0.85 to 0.5 when nodes become mobile (Figure 5.14). These results impact the message load which is therefore artificially low for 6Shot with mobility in low densities (Figure 5.16). In high densities (Figure 5.17) 6Shot performs as well in a mobile environment as in a static one. These results convey the fact that the DAG maintenance works well in high densities but

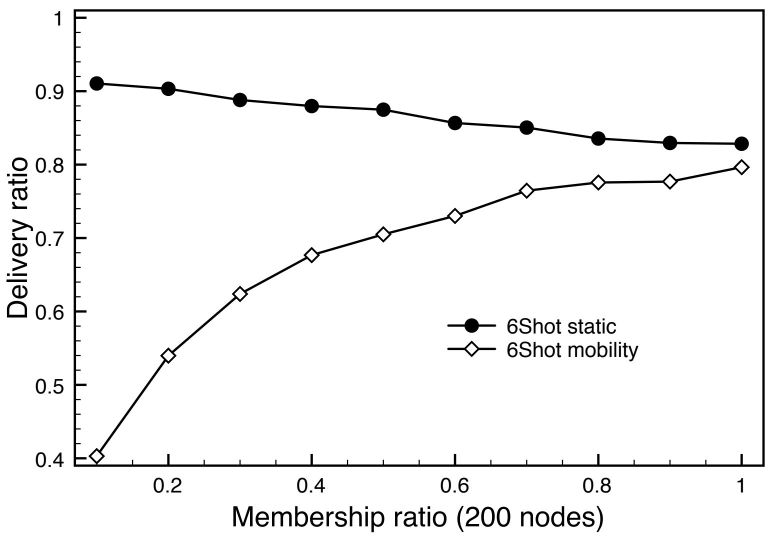


Fig. 5.14 Delivery ratio – with mobility.

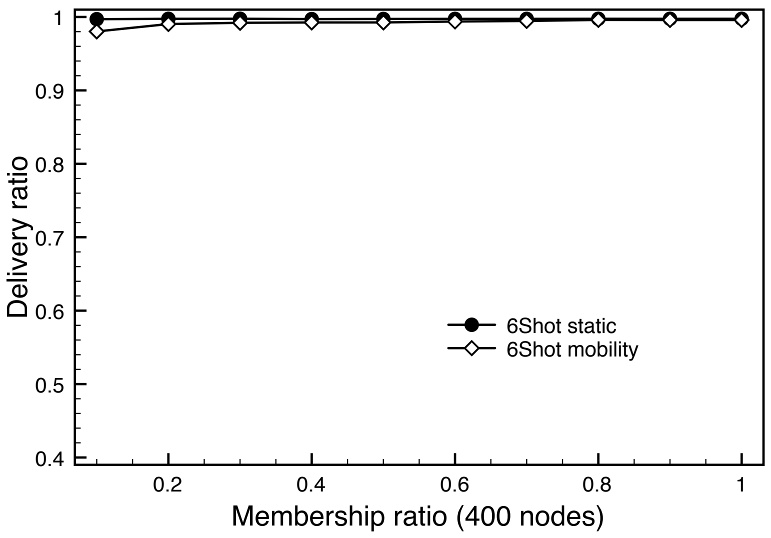


Fig. 5.15 Delivery ratio – with mobility.

is relatively vulnerable in low densities with a low group membership ratio. This is due to the fact that the smaller the number of members, the greater the chance to discard a message if the DAG is momentarily disrupted.

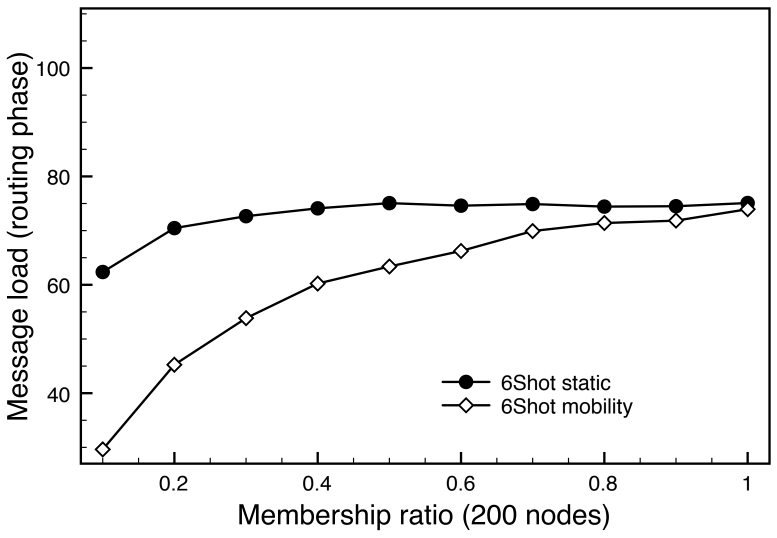


Fig. 5.16 Message load – with mobility.

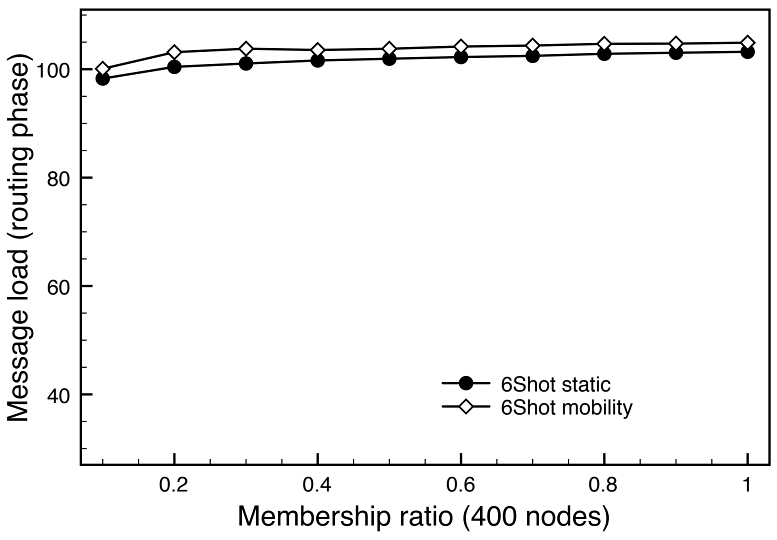


Fig. 5.17 Message load – with mobility.

5.5 Conclusion

Finding an efficient and generic multicast algorithm in a MANET is a challenging endeavor. In this paper we proposed 6Shot a context-aware multicast algorithm which exhibits a very efficient behavior in terms of message load from setting varying between small to very large proportions of multicast group members in the network, for a reasonable cost in terms of reliability. Future work will take into account the influence of errors in context sensors and we intend to perform a real life implementation of 6Shot on static wifi-enabled sensors.

References

[1] Gianpaolo Cugola and Matteo Migliavacca. A context and content-based routing protocol for mobile sensor networks. In *EWSN*, pages 69–85, 2009.

[2] E.M. Gafni and D.P. Bertsekas. Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology. *IEEE Transactions on Communications*, 29(1):11–18, 1981.

[3] B. Garbinato, A. Holzer, and F. Vessaz. Six-shot broadcast: a context-aware algorithm for efficient message diffusion in manets. In R. Meersman and Z. Tari, editors, *The 10th International Symposium on Distributed Objects, Middleware, Applications (DOA'08)*, pages 625–638. OTM 2008, Springer-Verlag, 2008.

[4] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999.

[5] Qing Huang, Yong Bai, and Lan Chen. Efficient lightweight broadcasting protocols for multi-hop ad hoc networks. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium*, pages 1 – 5, 2006.

[6] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the Workshop on Mobile Computing Systems and Applications. WMCSA'94*, 1994.

[7] Sung ju Lee, William Su, and Mario Gerla. On-demand multicast routing protocol.

[8] Ji Lusheng and M.S.Corson. A lightweight adaptive multicast algorithm. In *IEEE Global Telecommunications Conference, 1998. GLOBECOM 98.*, pages 1036–1042.

[9] Ewerton L. Madruga and J. J. Garcia luna aceves. Garcia-luna-aceves. scalable multicasting: The core-assisted mesh protocol. In *ACM/Baltzer Mobile Networks and Applications, Special Issue on Management of Mobility*, pages 151–165, 2001.

[10] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping 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, New York, NY, USA, 1999.

[11] Distributed Computing Group of ETH Zurich. Sinalgo: a simulation framework for manets. *http://dcg.ethz.ch/projects/sinalgo/*.

[12] Francisco Javier Ovalle-Martnez, Amiya Nayak, Ivan Stojmenovic, Jean Carle, and David Simplot-Ryl. Area-based beaconless reliable broadcasting in sensor networks. *IJSNet*, 1, 2006.

[13] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE Conference on Computer Communications, INFOCOM'97, April 7-11, 1997, Kobe, Japan*, volume 3. IEEE, April 1997.

[14] Vamsi Paruchuri, Arjan Durresi, and Raj Jain. Optimized flooding protocol for ad hoc networks. *CoRR*, cs.NI/0311013, 2003. informal publication.

[15] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99.*, pages 90–100. IEEE, 1999.

[16] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *ACM Mobicom*, pages 207–218, 1999.

[17] C. W. Wu and Y. C. Tay. Amris: A multicast protocol for ad hoc wireless networks. In *Proceedings of Military Communications Conference Proceedings, 1999. (MILCOM'99)*, pages 25–29. IEEE, 1999.

[18] Jason Xie, Rajesh R. Talpade, Anthony Mcauley, and Mingyan Liu. Amroute: ad hoc multicast routing protocol. *Mob. Netw. Appl.*, 7(6):429–439, 12 2002.

[19] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE.*

Chapter 6
Injecting Power-Awareness into Epidemic Information Dissemination in Sensor Networks

Published in
Future Generation Computer Systems
The International Journal of Grid Computing and eScience, Volume 26, Issue 6, June 2010.
Pages 868-876
Publisher: Elsevier
ISSN: 0167-739X

Abstract This paper presents and evaluates a novel approach to decrease the power consumption of epidemic information dissemination in sensor networks. In essence, our strategy consists in modulating the transmission range of sensors before they send messages. Since the range modulation follows a *power-law probability distribution*, we qualify our approach to information dissemination as being *power-law*. An obvious consequence of this strategy is that many nodes can reach few neighbors, while few nodes can reach many neighbors. To evaluate the effects of our approach, we inject the power-law range modulation into four existing epidemic algorithms and we compare their performances with their original versions, based on a fixed transmission range or on a uniform distribution of transmission ranges. This evaluation shows that our *power-law* approach improves the *efficiency* of the original algorithms in terms of power consumption, with no negative impact of their *effectiveness*, measured in terms of how many nodes have been reach after the dissemination.

6.1 Introduction

With the ubiquitous deployment of small embedded devices augmented with sensing capabilities, we are moving from rather static and centralized wire-based computer systems to much more dynamic distributed systems of heterogeneous sensing devices. Next-generation smart phones are good examples of such devices, as they start to commoditize sound and light sensors, positioning sensors, accelerometers, etc. These additional capabilities in turn open new perspectives in the development of context-aware applications running on ad hoc networks.

To take advantage of these emerging networks of sensing devices, one must basically be able (1) to *aggregate information* from its members, and (2) to *disseminate* *information* to its members, e.g., to redistribute collected information or even to ask members to execute individual/collective actions. At the physical level, the actual wireless technology being used determines how far each device can communicate and thus imposes constraints on the topology of the network. That is, a sensor can directly communicate only with a subset of the surrounding devices and must rely on these direct neighbors to communicate with more distant sensors. In addition, some devices might have heterogeneous communication capabilities: a sensor might use more than one technology and hence act as a gateway for other sensors. Moreover, devices are able to modulate their transmission power. Based on these low-level physical constraints, upper layers are then responsible for implementing information aggregation and dissemination protocols. At the heart of these protocols, one usually finds some kind of broadcast or multicast. In this paper, we focus on broadcast, which is the natural communication abstraction for information dissemination.

More generally, approaches to information dissemination can be categorized as either *unstructured* or *structured*. Unstructured approaches usually rely on some kind of probabilistic gossiping technique, also known as epidemic dissemination, either purely random or based on some smarter context-aware protocols, such as those presented in Section 6.2. Basically, gossiping consists in having each network sensor forward the messages it receives to a set of (more or less) randomly chosen neighbors. As a consequence, the paths followed by broadcasted messages are non-deterministic.

Such a gossip-based dissemination process can be seen as a simple *infection* process, in which infected sensors, i.e., those that received a message, may in turn infect with some probability their reachable neighbors. Because of their inherently decentralized nature, unstructured broadcasting solutions are good candidates for disseminating messages in wireless sensor networks.

Structured approaches, on the contrary, start off by building a virtual overlay network that exhibits a topology adequately supporting the information dissemination process. For this reason, structured approaches tend to require some state management and some coordination between distributed entities participating in the information dissemination process. While coordination might not necessarily be fully centralized, it usually implies the election of some global coordinator at least for some steps of the protocol, which can be rather resource consuming. In the context of sensor networks, since resources are limited in terms of battery, memory, processing, etc., structured approaches tend to be of limited interest.

The tradeoff between these two approaches is basically that of *effectiveness* vs. *efficiency*. In wireless sensor networks, the efficiency of a broadcast protocol is usually measured in terms of the average amount of resources required for broadcasting a message, whereas its effectiveness is usually measured as its average delivery ratio, i.e., the percentage of sensors that are actually reached on average. In the following, we sometime use the term *reliability* instead of effectiveness, as the latter indeed measures how reliable a particular dissemination protocol is in reaching all nodes of the network.

Intuitively, structured approaches tend to favor reliability, because they aim at devising topology control schemes that maximize the delivery ratio. Conversely, unstructured approaches tend to be decentralized, and hence favor efficiency, as they do not need to build or maintain an overlay topology, and require no coordination between sensors and hence consume fewer resources.

6.1.1 Contribution

In this paper, we propose an unstructured and decentralized approach to modulating the transmission range of sensors, allowing for energy optimization and no loss in reliability. Intuitively, our approach consists in having sensors execute a traditional broadcasting algorithm but additionally modulate the power of their wireless transmissions according to a *power-law distribution*. That is, whenever a sensor has to wirelessly transmit a message in its neighborhood, it randomly draws the transmission power it will use from that distribution.

Because the power-law distribution is given a priori to each sensor, our protocol does not add centralized or stateful behavior to existing algorithms. That is, if an algorithm is stateless and decentralized, such as the gossip protocol, it remains decentralized and stateless after injecting transmission range modulation. However, the power-law modulation of wireless transmissions produces, for each broadcast, an actual routing topology that exhibits characteristics similar to a scale-free overlay network, which has been observed to be highly efficient for information transmission in other communication contexts.Since the system is deployed in a metric space, geographical constraints play an important role in the actual emerging topology. For example, scale-free networks are difficult to embed in two-dimensional space [9] but network degree distributions with long tails can nevertheless be obtained, as shown in [8]. As we shall see, this approach allows us to save power, with no negative impact on effectiveness.

6.1.2 Related Work

In the past few years, information dissemination protocols have been extensively studied in the context of wireless sensor networks, with the goal to propose power-efficient solutions [2, 10, 18, 15]. In this context, it is a known experimental results that communication is expensive compared to processing when it comes to energy consumption [20]. That is, in terms of energy, transmitting a single bit is roughly equivalent to processing a thousand operations in a typical sensor node [19]. Furthermore, several short-range transmissions can be less energy costly than one long-range transmission [23], meaning that multi-hop communication is very interesting when it comes to saving energy in wireless sensor networks [19].

Following the taxonomy of energy saving techniques proposed in [15], our work can be categorized as relying on *no energy information*, on *multi-hop communication* and on both *local* and *localized processing*. The *no energy information* assumption simply means that nodes do not know how much energy is at their disposal. Then, some of the algorithms we discuss in this paper do not rely on any coordination between nodes (*local processing*), whereas other algorithms rely on limited coordination between nodes and their direct neighbors (*localized processing*).

In parallel to these researches, there have been several suggestions as to how to use bio-inspired techniques in (possibly mobile) ad hoc networks [1, 22, 26]. Also, in the wider area of Grid computing, bio-inspired techniques such as ant-based algorithms have been successfully employed to solve complicated tasks such as balanced job scheduling [3]. Since all the algorithms we examine hereafter rely on some kind of epidemic dissemination, and how such a dissemination behaves in a typical small-world population, our research can be seen as applying bio-inspired techniques to the problem of disseminating information in wireless sensor networks.

Interestingly, few of these researches tried to explicitly take advantage of small-world topologies, and especially scale-free ones, in order to efficiently disseminate information in wireless sensor networks, although the latter have been found to be efficient in several studies. In [24] for instance, the authors discuss their model of information dissemination and membership management in the context of a heterogeneous communication network consisting of computing sensors, local high-speed links and wide-area links. Interestingly, their networks are small worlds but they are not scale-free. Along the same line, the authors of [13] propose a gossip-based protocol for aggregating information in a peer-to-peer manner. When simulating their protocol on various topologies, they found out that performance is independent of network size but highly sensitive to the topology. That is, their gossip protocol is efficient on topologies that have a small diameter, in particular on small-world topologies. This concurs with our own findings that scale-free topologies are beneficial to the performance of gossiping protocols [5]. In [4], the authors employ a bio-inspired multi-agent approach to spatially sort and discover information about the resources offered by a Grid. Agents, whose behavior is inspired by ant colonies, replicate and distribute resource descriptors according to the class to which the corresponding resources belong and they use an epidemic diffusion mechanism related to our approach. In wireless sensor networks, a key issue consists in deploying the most effective topology for a given problem, either through physical sensor placement or through logical topology control. For instance, the authors of [21] propose connection protocols approximating scale-free topologies on the sole basis of local information exchange, in order to improve efficiency and robustness of pervasive networks. In [11] the author investigates the opportunity to take advantage of small-world topologies to build efficient protocols for wireless networks. In peer-to-peer systems, a Watts–Strogatz type small-world architecture has been successfully employed for content discovery by creating and using long-range shortcuts toward remote peer groups [14]. Two other works are based on Watts–Strogatz type networks [25], which are more difficult to embed in physical space than scale-free networks.

6.1.3 Roadmap

The remainder of this paper is organized as follows. Section 6.2 briefly introduces the algorithms used to evaluate our approach, while Section 6.3 specifies our model and describes our power-law approach in the context of information dissemination protocols. Sections 6.4 and 6.5 then present and discuss the performance evaluation of our approach. Finally, Section 6.6 places our results into perspective and discusses ongoing and future work.

6.2 Broadcasting algorithms

The four epidemic dissemination algorithms in which we will inject a power-law modulation of the transmission range are representative of various forwarding techniques. Apart from the probabilistic gossip, all can be considered to be *context-aware*. That is, they rely on information about the context in which they execute, such as message traffic or node location, to optimize their behavior.

6.2.1 Probabilistic gossiping

The *Probabilistic Gossiping (PG)* algorithm [16] works similarly to simple flooding [10] except that sensors broadcast a received message only with a certain probability p. Simple flooding can be seen as a special case of PG where $p = 1$. In PG, efficiency can be increased by reducing the probability p, which determines the forward ratio, but as the efficiency increases the reliability decreases. The magnitude of the decrease in reliability depends on the density of the network.

6.2.2 Counter-based scheme

In the *Counter-Based Scheme (CBS)* [16], when a sensor receives a new message m, it fixes a random waiting delay before making the forwarding decision. During this delay, the sensor counts the number of retransmission of m it receives. After the waiting delay has elapsed, the message is only forwarded if the number of retransmissions is smaller than a predetermined *threshold*. As a result, the forwarding ratio depends on the sensor density, i.e., in low density areas this ratio will be high, whereas in high density areas it will be low. This simple algorithm turns out to be very competitive as it does not concede much reliability for highly improved efficiency.

6.2.3 Hop-count aided broadcasting

The *Hop-Count Aided Broadcasting (HCAB)* [12] scheme is similar to *CBS* except that each message has a *hop counter*, which starts at 0 and is incremented after each hop. This creates geographical zones within which message retransmissions have the same hop count. When a sensor receives a message for the first time with a hop counter set to a value v, then alike in the *CBS* scheme the sensor sets a waiting delay and listens to message retransmissions. The main differences with *CBS* is that only retransmissions with a hop counter greater than v are taken into account, and when a message is forwarded its hop counter is incremented. This algorithm aims at being more reliable than *CBS* at the expense of increased message redundancy.

6.2.4 Six-shot broadcast

The *Six-Shot Broadcast (6SB)* [6] algorithm is very similar to CBS (Section 6.2.2). The key difference lies in the fact that each sensor determines its waiting delay based on its distance to six predetermined targets, as shown in Figure 6.1, rather than choosing the delay randomly.

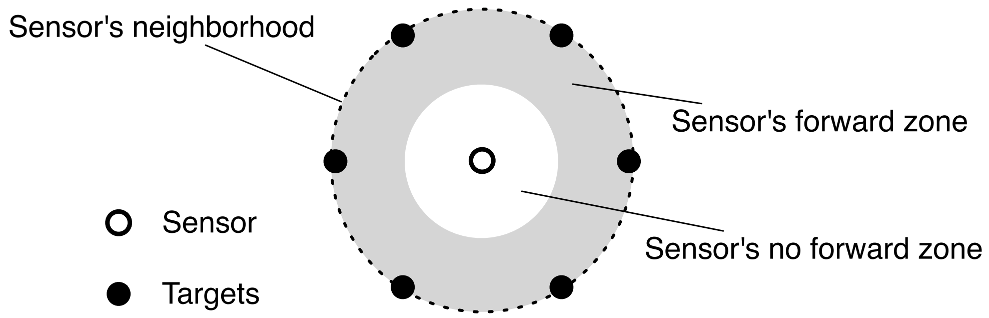


Fig. 6.1 Six-shot broadcast principles

More precisely, the idea is to compute the waiting delay according to the distance of the closest target: the closer the target, the shorter the delay. In order to compute these targets, the location of the sender or the forwarder is embedded into the message. In addition, only sensors located in the so-called *forward zone* may resend, while sensors located in the *no forward zone* remain quiet.

6.2.5 Summary of the algorithms

We can categorize these algorithms via three dimensions, namely the *context type*, *message content*, and *delay function*. Table 6.1 summarizes the key differences between these algorithms using these dimensions.

The *context type* characterizes the type of context information used by the algorithm to optimize its behavior. PG algorithm is *context oblivious* because it requires no information. CBS and HCAB are *traffic aware* because they have a waiting time during which they monitor the traffic on the network. With 6SB, each sensor is required to know its own location[1]. Of course, the context type has an impact on the *message content*, possibly on the *delay function*. That is, the delay can either be random or can be computed as a function of some context information, e.g., the sensor location.

Algorithm	Context Type	Message Content	Delay Function
PG	context oblivious	data	no delay
CBS	traffic aware	data	random
HCAB	traffic aware	data + counter	random
6SB	location aware	data + sender position + range	location dependent

Table 6.1 Classification of dissemination algorithms

[1] Typically via some GPS capability.

6.3 Model definition and approach

We model a wireless sensor network as a set of distributed processes communicating via low-level broadcasting. The system is considered to be asynchronous, meaning that we make no assumption about the time it takes for sensors to execute and for messages to be transmitted. In addition, each sensor can modulate its transmission power. In the following, to avoid any confusion with the notion of *power-law* graph, we say that sensors modulate their *transmission range* rather than their transmission power, the former being a function of the latter.

Formally, the system's topology is defined by a graph $G = (\Pi, \Lambda)$, where $\Pi = \{n_1, n_2, ...\}$ is a set of processes ($|\Pi| = n$), and $\Lambda = \{l_1, l_2, ...\} \subseteq \Pi \times \Pi$ is a set of *directional* communication links. A link from n_i to n_j is also denoted by $l_{i,j}$. If $l_{i,j} \in \Lambda$, we say that n_j is neighbor of n_i. The set of all n_i's neighbors is denoted by $neighbors(n_i)$; $k = |neighbors(n_i)|$ denotes the out-degree of a process. The *graph degree distribution* function $P(k)$ gives the probability that a randomly selected vertex has degree k.

6.3.1 Power-law approach

In essence, our so-called *power-law approach* consists in having each sensor modulate its transmission range using a power-law distribution function, in order to disseminate information through a network with properties similar to those of a scale-free network. That is, as suggested by Figure 6.2, most sensors will thus choose a short range, while few sensors will choose a long range.

To evaluate our approach, we also test a simpler way to modulate the transmission range, which consists in selecting the range using a uniform probability, i.e., a long range has the same probability to be selected as a short range. It is important to understand that the modulation of transmission ranges creates the possibility to have uni-directional links between sensors, i.e., it is possible that Sensor A can reach Sensor B but Sensor B is unable to reach Sensor A (because it chose a shorter transmission range). Our power-law approach is thus implicitly building directed networks. A thorough study of the impact of using an uniform probability or a power-law probability can be found in [7].

6.3.2 Distribution function

Technically speaking, the distribution function returns a value between a minimum strictly greater than 0, which depends on the density of sensors in the field, and a maximum value given by constraints imposed by the wireless technology. Note

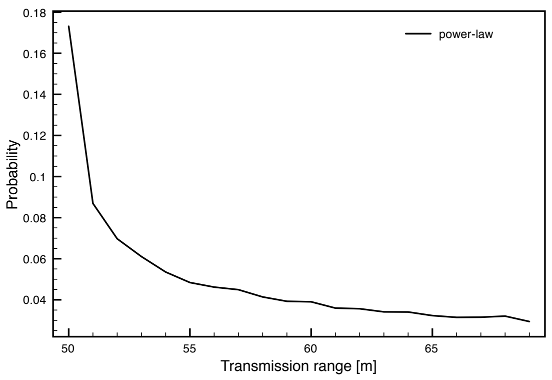


Fig. 6.2 Power-law function distribution. The function is discrete, the continues curve is a guide for the eye.

that sensors compute a different transmission range each time a message has to be forwarded. Typically, a sensor using a long transmission range at some point will not use this range for its entire lifetime, because its battery would run out very quickly.

Hereafter, Equation 6.1 defines such a power-law distribution function f. In this equation, c_1, c_2 are constants, while x is a randomly chosen value, and α typically lies between 2.0 and 3.0 and defines the shape of the power-law distribution.

$$f(x, \alpha) = c_1 + c_2 \times x^{-\alpha} \tag{6.1}$$

6.3.3 Generic algorithm structure

Algorithm 6.13 provides the generic structure for applying our power-law approach to the four algorithms presented in Section 6.2. This generic algorithm is composed of three main phases: the *initialization* (lines 1-4), the *initial broadcast* (lines 5-8) and the *processing of received messages* (lines 9-15).

The *initialization* phase calls the init() function of the broadcast algorithm (PG, CBS, HCAB or 6SB) and defines variable *range* and constant *alpha*. The *initial broadcast* is implemented in primitive broadcast(msg), which uses the maximum technical range for the initial send.[2]

Each time a message is received by the sensor, it is passed to the alreadyReceived() function, which checks whether the message was already received. If not, the message is delivered to the sensor via the deliver() callback. Then, the shouldForward() function is called to check whether the message should be retransmitted or not (this

[2] This minor initial bias allows to dramatically reduce latency and increase reliability at the same time.

function is typically dependent on the actual algorithm to which our power-law approach is being applied). If so, the variable *range* is set to a new value, using the power-law function $f(x, \alpha)$ of Equation 6.1, and the message is forwarded.

```
1: Initialization:                                        {all sensors}
2:     init()
3:     $\alpha \leftarrow ...$
4:     $range \leftarrow \perp$

5: To execute broadcast($data$) do:                       {initial sender}
6:     $msg.data \leftarrow data$
7:     $range \leftarrow maxRange$
8:     mac-broadcast($msg, range$)

9: on mac-deliver($msg$):                  {each time a message is received}
10:    if $\neg$ alreadyReceived($msg$) then
11:        deliver($msg.data$)
12:        if shouldForward($msg$) then
13:            $x \leftarrow$ random()
14:            $range \leftarrow f(x, \alpha)$                {Equation 6.1}
15:            mac-broadcast(msg,range)
```

Algorithm 6.13 Generic algorithm structure

6.4 Performance evaluation

In the following, we evaluate and discuss the performance of our approach in terms of delivery ratio and of forward ratio, the latter being seen as functions of time and energy consumption.

More precisely, we define the *delivery ratio* as the number of sensors that have received the message after dissemination is over, with respect to the total number of sensors. The *forward ratio* corresponds to the number of sensors that forward the message; in the literature, this value is sometimes assimilated to the number of messages being sent and typically corresponds to the amount of communication generated by the algorithm. In our simulations, we make the assumption that ten rounds are made in one second. Each forward is done in one round. The *energy consumption* is then given in Joules and represents the power necessary to execute a broadcast. The energy consumption of one message is equal to d^{β} where d is the transmission distance and β the path loss exponent ($2 \leq \beta \leq 4$)[23]. We use $\beta = 4$ for our performance evaluation, which simulates a dense network with a lot of interferences.

Our models were simulated with the *Sinalgo* tool [17]. *Sinalgo* is a simulation framework written in Java, which allows to create complex distributed systems with discrete time events. In the following figures, each curve is the average of 100 independent executions. We generate 100 different random networks and we use this same set with each algorithm. The fixed parameters are the probability to gossip (0.8), the map size (700 meters $\times$ 700 meters) and the number of sensors (800, which implies a density of 16.3 sensors/hm^2). In all simulations, we use a two-dimensional surface with periodic conditions, which yields a torus. Sensors are deployed randomly according to an uniform probability to set its position. We use two different distribution functions between 50 meters and 70 meters: a power-law distribution with a coefficient α equals to 2.6 and an uniform distribution with coefficient α equals to 0.[3] We run the simulations with the following algorithms adapted to the variable range approach: probabilistic gossip (PG), counter based scheme (CBS), hop-count aided based (HCAB) and six-shot broadcast (6SB). Table 6.2 summarizes these parameters.

Parameters	Values
Map Size	$700\ m \cdot 700\ m$
Map area	49 *ha*
Min. transmission range	50 *m*
Max. transmission range	70 *m*
Number of nodes	400 or 800
Number of sender	1
Number of simulations	100
Power-law α coef.	2.6 or 0

Table 6.2 Simulations parameters

6.4.1 Injecting modulation of the transmission range

Figure 6.3 shows global results for the delivery ratio as a function of the energy consumption. The three dash-dot curves represent the performance of the PG algorithm with various strategies for choosing the transmission range (fixed range, uniform distribution and power-law distribution); we can see that all curves reach almost 1 in delivery ratio. Relatively to the energy consumption however, the strategy with a fixed transmission range is clearly the worst. For example, to reach a 0.9 delivery ratio, the fixed range strategy requires 1.4×10^{10} Joules, while the uniform distribution strategy requires 0.8×10^{10} Joules. The strategy based on our power-

[3] By uniform distribution we mean that the technical range is uniformly distributed between the minimum and the maximum technical range, which obtains with $\alpha = 0$.

law approach yields the best results, as it only needs 0.6×10^{10} Joules for the same delivery ratio.

We can also see that the delivery ratio is quite high for all algorithms and all strategies, which is due to the dense network we created for our simulation. Not surprisingly, the PG algorithm is clearly the worst algorithm compared to CBS, HCAB and 6SB, so we detail the performance of these other algorithms hereafter.

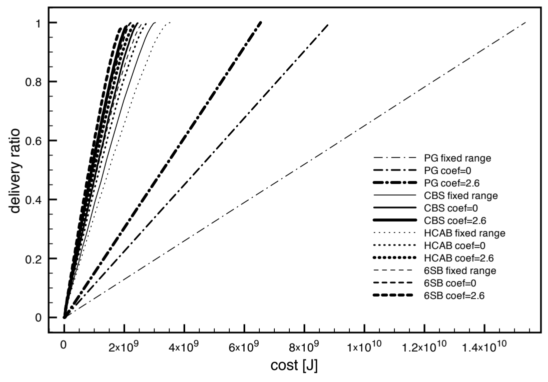


Fig. 6.3 Delivery ratio of all algorithms as a function of cost (energy consumption)

In Figure 6.4, the curves depict results for the CBS algorithm and show that for a defined amount of energy, the best strategy is the one based on our power-law approach. For instance, with a number of Joules equals to 2×10^9, our power-law approach has a delivery ratio equals to 0.95, while the delivery ratio is only 0.87 for the uniform strategy and even less for the fixed range strategy (0.75).

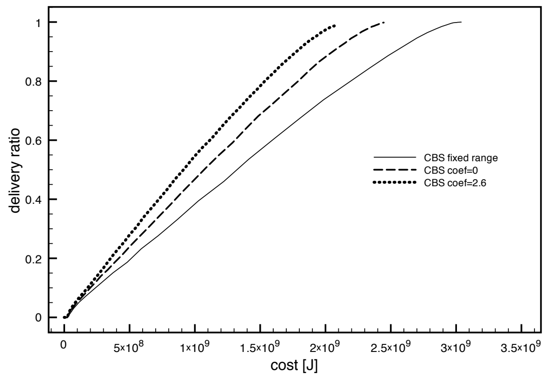


Fig. 6.4 Delivery ratio of CBS algorithm as a function of cost (energy consumption)

For the HCAB algorithm (Figure 6.5) and the 6SB algorithm(Figure 6.6), results are similar. We can observe that 6SB performs best, followed by CBS, HCAB and PG.

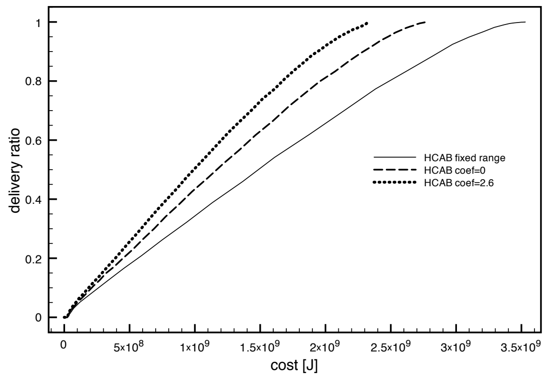


Fig. 6.5 Delivery ratio of HCAB algorithm as a function of cost (energy consumption)

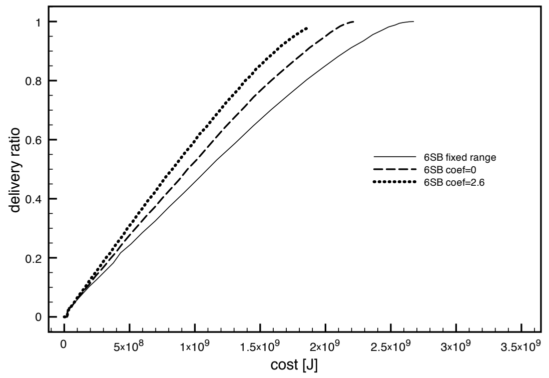


Fig. 6.6 Delivery ratio of 6SB algorithm as a function of cost (energy consumption)

6.4.2 Delivery ratio and latency

Figure 6.7 depicts the delivery ratio for the different algorithms as a function of time. For sake of clarity, only results for the power-law strategy and for the uniform strategy are given here. Results show that using a power-law instead of an uniform probability slows down the dissemination. For each algorithm, curves obtained using a power-law distribution are at the right of the curves of the algorithms using a uniform probability, meaning that they are slower. We can thus classify algorithms by according to their respective latency: the fastest algorithm being the PG, which is also the simplest, then HCAB, CBS and 6SB.

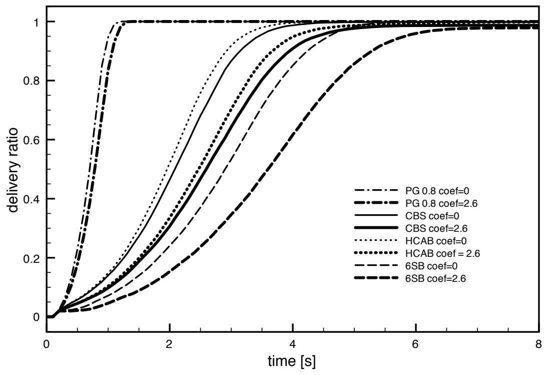


Fig. 6.7 Delivery ratio of the algorithms as a function of time.

6.4.3 Forward ratio and latency

For a better view of the results, we present hereafter two graphics: Figure 6.8 *with* the PG algorithm, and Figure 6.9 *without* the PG algorithm, as it has a much higher forward ratio than all the other algorithms. Apart from the PG algorithm, which has a 0.8 forward ratio for both the uniform strategy and the power-law strategy, all the other algorithms exhibit a significant difference between these two strategies.

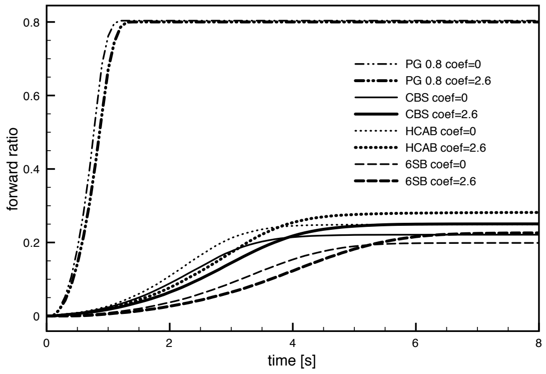


Fig. 6.8 Forward ratio of the algorithms as a function of time.

Figure 6.9 focuses on this difference. At the beginning, the power-law strategy forwards less messages than the uniform strategy. This finding can explain the relative slowness of the power-law strategy discussed in Section 6.4.2. After some time however, power-law strategy uses more forward than the uniform strategy. This explains why the delivery ratio is, at the end of the execution, is equal in both strategies. The ranking based on latency is obviously correlated with the number of forwards needed. We find the same ranking as in Section 6.4.2: PG, HCAB, CBS and 6SB.

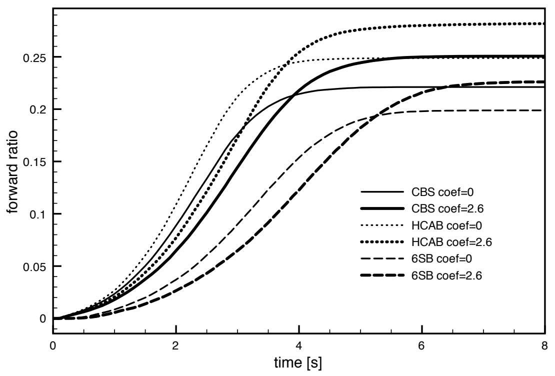


Fig. 6.9 Detailed forward ratio as a function of time.

6.4.4 Energy consumption

The energy consumption (or *cost*) is defined as the energy needed to execute the algorithm. Figure 6.10 shows that the PG algorithm is the less power-efficient algorithm, i.e., its low latency has a non-negligible price. Nevertheless, we can observe that broadcasting with an uniform strategy gives a total cost around 8.8×10^9 Joules and a power-law strategy has a total cost of 6.6×10^9 Joules, a gain of 25% in energy consumption. Other algorithms use approximately 2×10^9 Joules, which is a lot lower than the PG algorithm.

More precisely, Figure 6.11 shows that for each algorithm, the power-law strategy is more battery-efficient than the uniform strategy. The ratio is less than the one of the PG algorithm but the gain is far from being negligible. For HCAB, the gain is around 15%, for CBS around 17% and for 6SB around 16%. This is probably due to the fact that there is less headroom for improvement with these algorithms than with PG. Overall, the three context-aware algorithms reduce the PG power consumption by 75%. So for a small difference in latency, we can obtain quite a big difference in power consumption. Based on their energy consumption, the four dissemination algorithms are now ranked in opposite order: first 6SB, then CBS and HCAB, and finally PG.

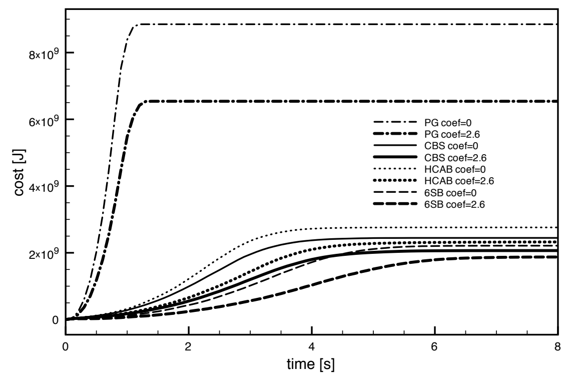


Fig. 6.10 Energy cost as a function of time.

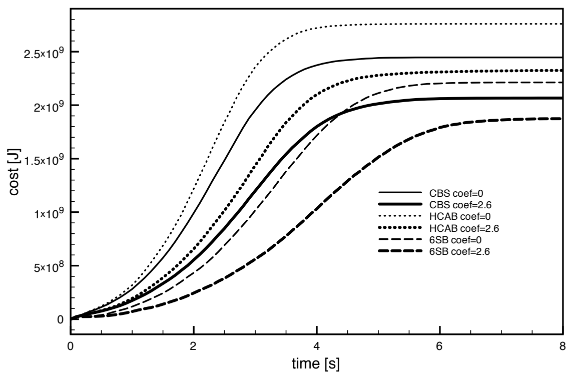


Fig. 6.11 Detailed energy cost as a function of time.

6.5 Analysis and discussion

In this section, we further analyze and discuss two key aspects of our performance evaluation. In section 6.5.1, we discuss the impact of sensors density on the different dissemination algorithms, in the light of the various strategies used to modulate the transmission range. In section 6.5.2, we then show that our power-law approach offers a very interesting balance between efficiency and effectiveness.

6.5.1 Low densities

Throughout Section 6.4, we presented results for a density of 16.3 sensors/ha^2 (800 nodes in a $700m \times 700m$ area). This is a density that gives strong guarantees that every sensor in our network is connected to the giant component. To illustrate the behavior of the different range modulation strategies in lower densities, we ran simulations with 400 nodes (8.2 sensors/ha^2). This density is just sufficient to ensure a good probability that sensors are connected to the giant component, when using a minimum technical range of $50m$.

Figure 6.12 shows the results of different modulation strategies for the CBS algorithm in low densities.[4] Figures 6.13 and 6.14 illustrates the same for HCAB and 6SB respectively.[5] Compared to the results presented in Section 6.4, these new figures additionally contain the curves corresponding to a fixed transmission range set to the minimum ($50m$), not only curves corresponding to the maximum range ($70m$) as presented in Section 6.4. Impact of using a minimum technical range will be discuss more in details in Section 6.5.2.

Coming back to low densities, for every algorithm, the results are basically the same as for higher densities: the power-law strategy has the lower cost followed by the uniform strategy and finally the (maximum) fixed range. The only impact of low densities is that strategies based the minimum technical range cannot approach a delivery ratio of 1. In fact, the more sensors are using a minimum technical range, the lowest the delivery ratio. So the strategy with the lowest delivery ratio is the minimum fixed range, followed by the power-law strategy, the uniform strategy and finally the maximum fixed range strategy.

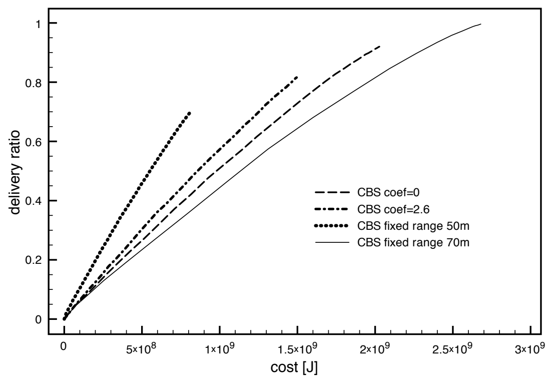


Fig. 6.12 Delivery ratio of CBS algorithm as a function of cost (energy consumption) in low densities (400 nodes).

6.5.2 Balancing efficiency and effectiveness

While in Section 6.4, we claimed that our power-law strategy should be preferred when it comes to energy consumption, Figures 6.12 to 6.14 seem to contradict this

[4] See Figure 6.4 for results in high densities.

[5] See Figures 6.5 and 6.6 for results in high densities.

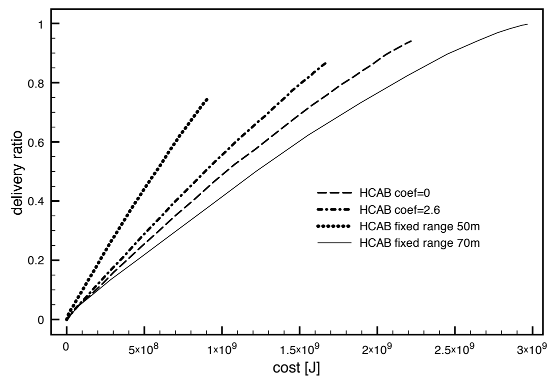


Fig. 6.13 Delivery ratio of HCAB algorithm as a function of cost (energy consumption) in low densities (400 nodes).

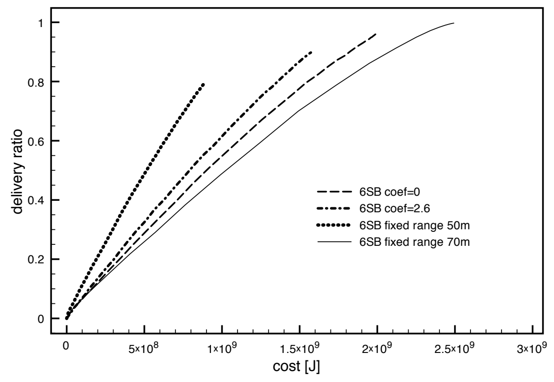


Fig. 6.14 Delivery ratio of 6SB algorithm as a function of cost (energy consumption) in low densities (400 nodes).

claim, as they show a lower energy consumption when using the minimum transmission range than when using our power-law modulation strategy. Note that this is not only true in lower densities but also in high densities, as depicted in Figures 6.15 to 6.18. That is, for each dissemination algorithm, the minimum fixed range strategy is the best in terms of energy consumption, followed by the power-law strategy and then the maximum fixed range strategy. Given these results, a legitimate question is the following: *why not simply use the minimum transmission range instead of using our power-law strategy?*

The problem with a strategy consisting in systematically using the minimum transmission range lies in the uncertainty regarding the density of sensors and their topology. For this reason, it is very difficult to safely determine what is the minimal fixed range that should be used to optimize the energy consumption and to ensure the delivery of messages to the whole network. For instance, results of Section 6.5.1 show that using a fixed transmission range of $50m$ (the minimal range) is certainly efficient in terms of energy consumption but it also dramatically decreases the delivery ratio (from almost 1 to about 0.8). In other words, this strategy cannot safely ensure both *efficiency* and *effectiveness*.

Our power-law strategy, on the other hand, offers an interesting middle ground between these two goals: it allows us to save energy (efficiency) without compromising the delivery ratio (effectiveness). This is clearly visible in Figures 6.15 to 6.18: the results of the power-law strategy alway lies between the fixed maximum range strategy and the the fixed minimum range. And since the results of Section 6.4 also show that the power-law strategy is better than the uniform distribution strategy, we believe that our approach offers a very good balance of efficiency and effectiveness, when it comes to disseminate information in sensor networks.[6].

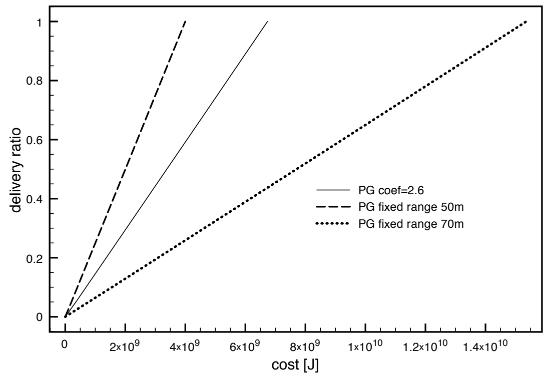


Fig. 6.15 Delivery ratio of PG 0.8 algorithm as a function of cost (energy consumption).

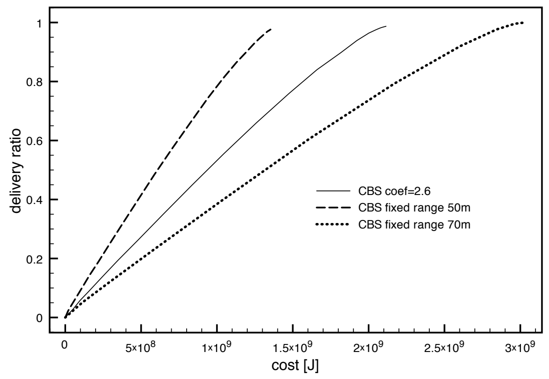


Fig. 6.16 Delivery ratio of CBS algorithm as a function of cost (energy consumption).

[6] Latency is another minor advantage of the power-law strategy over the minimum fixed range

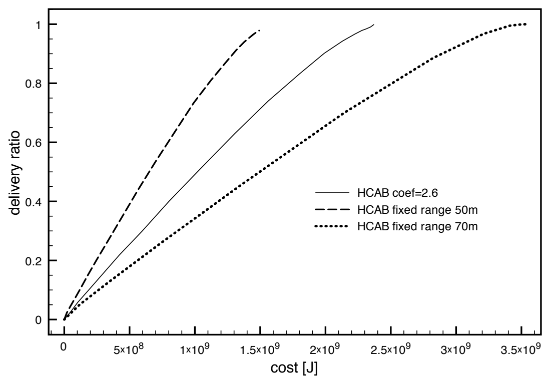


Fig. 6.17 Delivery ratio of HCAB algorithm as a function of cost (energy consumption).

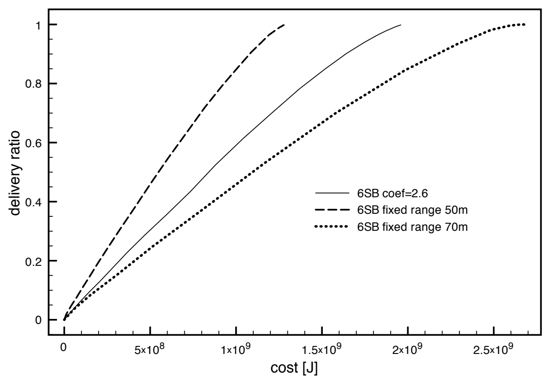


Fig. 6.18 Delivery ratio of 6SB algorithm as a function of cost (energy consumption).

6.6 Concluding Remarks

In this paper, we introduced a novel technique to decrease the energy consumption when disseminating information in sensor networks. Intuitively, our technique consists in injecting power-awareness into existing epidemic dissemination protocols, by modulating the transmission range of sensors when they broadcast messages. This modulation follows a power-law probability distribution, so we qualify our approach as being *power-law*. Performance analysis shows that our approach increases *efficiency* (measured in terms of energy consumption) with virtually no negative impact on *effectiveness* (measured in terms of delivery ratio). This is due to the fact that the network organizes itself as a small-world, where few sensors act as hubs while most sensors rely on multi-hop communication for disseminating information. Our approach, being local and stateless, is easily extensible to mobile devices. Current ongoing work precisely involves adding simple mobility schemes and failures to the picture, and further improving our power-law gossiping approach using location information about sensors.

References

[1] E. Alba, J.García-Nieto, J. Taheri, and A. Zomaya. New research in nature inspired algorithms for mobility management in (gsm) networks. In *Applications of Evolutionary Computing*, volume 4974 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2008.

[2] U. Cetintemel, A. Flinders, and Y. Sun. Power-efficient data dissemination in wireless sensor networks. In *MobiDe '03: Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, pages 1–8, New York, NY, USA, 2003. ACM Press.

[3] Ruay-Shiung Chang, Jih-Sheng Chang, and Po-Sheng Lin. An ant algorithm for balanced job scheduling in grids. *Future Generation Computer Systems*, 25(1):20 – 27, 2009.

[4] A. Forestiero, C. Mastroianni, and G. Spezzano. Reorganization and discovery of grid information with epidemic tuning. *Future Gener. Comput. Syst.*, 24(8):788–797, 2008.

[5] B. Garbinato, D.Rochat, and M. Tomassini. Impact of scale-free topologies on gossiping in ad hoc networks. In *NCA*, pages 269–272, 2007.

[6] B. Garbinato, A. Holzer, and F. Vessaz. Six-shot broadcast: A context-aware algorithm for efficient message diffusion in manets. In *OTM '08: Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I on On the Move to Meaningful Internet Systems:*, pages 625–638, Berlin, Heidelberg, 2008. Springer-Verlag.

[7] B. Garbinato, D. Rochat, and M. Tomassini. Des réseaux non-homogènes dans les réseaux ad hoc. In *NOTERE '09: Proceedings of the 9th Annual International Conference on New Technologies of Distributed Systems*, pages 136–143, 2009.

[8] B. Garbinato, D. Rochat, and M. Tomassini. Power-efficient epidemic information dissemination in sensor networks. In *BADS '09: Proceedings of the 2009 workshop on Bio-inspired algorithms for distributed systems*, pages 69–76, New York, NY, USA, 2009. ACM.

[9] Y. Hayashi. A review of recent studies of geographical scale-free networks. *IPSJ Trans. Special Issue on Network Ecology*, 47(3):776–785, 2006.

[10] W.R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM Press.

[11] A. Helmy. Small worlds in wireless networks. *Communications Letters, IEEE*, 7(10):490–492, 2003.

[12] Q. Huang, Y. Bai, and L. Chen. Efficient lightweight broadcasting protocols for multi-hop ad hoc networks. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium*, pages 1 – 5, 2006.

[13] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems (TOCS)*, 23(3):219–252, 2005.

[14] Lu Liu, Nick Antonopoulos, and Stephen Mackin. Fault-tolerant peer-to-peer search on small-world networks. *Future Generation Computer Systems*, 23(8):921 – 931, 2007.

[15] R. A. F. Mini and A. A. F. Loureiro. *Energy in Wireless Sensor Networks*, pages 3–540. 2009.

[16] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. 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, New York, NY, USA, 1999. ACM.

[17] Distributed Computing Group of ETH Zurich. Sinalgo: a simulation framework for manets. http://dcg.ethz.ch/projects/sinalgo/.

[18] L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti. Localized techniques for broadcasting in wireless sensor networks. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 41–51, New York, NY, USA, 2004. ACM Press.

[19] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, May 2000.

[20] V. Raghunathan, C. Schurgers, Sung Park, and M.B. Srivastava. Energy-aware wireless microsensor networks. *Signal Processing Magazine, IEEE*, 19(2):40 –50, mar 2002.

[21] F. Saffre, H. Jovanovic, C. Hoile, and S. Nicolas. Scale-free topology for pervasive networks. *BT Technology Journal*, 22(3):200–208, 2004.

[22] M. Saleem and M. Farooq. BeeSensor: a bee-inspired power aware routing protocol for wireless sensor networks. In *Applications of Evolutionary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 81–90. Springer, 2007.

[23] Zach Shelby, Carlos Pomalaza-R°ez, Heikki Karvonen, and Jussi Haapola. Energy optimization in multihop wireless embedded and sensor networks. *International Journal of Wireless Information Networks*, 12:11–21, 2005.

[24] S. Voulgaris, M. Jelasity, and M. van Steen. A robust and scalable peer-to-peer gossiping protocol. In *Agents and Peer-to-Peer Computing*, volume 2872 of *Lecture Notes in Computer Science*, pages 47–58. Springer, 2004.

[25] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[26] H. F. Wedde, C. Timm, and M. Farooq. BeeHiveGuard: a step towards secure nature inspired routing algorithms. In *Applications of Evolutionary Computing*, volume 3907 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2006.

Chapter 7
PLAN-B: Proximity-based Lightweight Adaptive Network Broadcasting

Published in the proceedings of the 10th IEEE International Symposium on Network Computing and Applications (NCA)
Cambridge, MA, USA, August 25-27, 2011
Publisher: IEEE Computer Society
ISBN: 978-0-7695-4489-2

Abstract Broadcast is an important building block in ad hoc networks. Its challenge is to deliver a message to all nodes in the network for a reasonable cost in terms of message load and delay. Several context-aware broadcasting protocols have been proposed in order to meet this challenge, using location or proximity information in order to fine-tune retransmission decisions. However, existing protocols often target one specific setting and can reveal to be sub-optimal when settings change. Typically, optimal parameters for dense networks will differ from optimal parameters for sparse networks. To address this issue, we propose PLAN-B an adaptive proximity-based broadcast protocol that offers the possibility to define policies in order to adapt its parameters for different network settings at runtime. Our performance evaluations show that PLAN-B outperforms existing static and adaptive protocols in terms of message load in changing and unknown densities up to a factor of 2.

7.1 Introduction

With the widespread usage of smart mobile devices and the perspective of ever increasing connectivity decentralized wireless architectures on the network layer, such as mobile ad hoc network or sensor networks, will play an increasingly important role. Such architectures bring about many new challenges as well as opportunities in distributed computing systems, in particular in the realm of communication protocols.

In a mobile ad hoc network (MANET), when a message m is sent by a node n, all nodes within the n's transmission range receive m. The *transmission range* of a node is the area that is covered by its radio signal. This area varies depending on the communication technology used, e.g., Bluetooth up to 10 m, WiFi up to 250 m, and on the obstacles between nodes. This mechanism is managed by the nodes' MAC layer. The *neighborhood* of a node is composed of all nodes in its transmission range. In order to send a message to nodes outside of one's neighborhood, whether the message is destined to one node (unicast), several nodes (multicast) or all nodes (broadcast), some other nodes must retransmit the message until it reaches its destination. In this case, the network is said to be *multi-hop*, a hop being the link between a node to another one in its transmission range. Because of their mobility, devices in decentralized networks usually have limited resources such as battery power and bandwidth. So protocols have to be optimized for such resource limitations. An additional issue is the influence of execution environments on the performance of protocols. For instance different levels of network density or the availability of sensors can dramatically change the outcome of a performance evaluation. In order to overcome these issues, we present PLAN-B, an adaptive proximity-based broadcasting protocol for decentralized ad hoc networks.

This paper is structured as follows. Section 7.2 presents the broadcast problem and related work. Then, Section 7.3 presents PLAN-B, a novel adaptive proximity-based broadcasting protocol. Section 7.4 presents the settings of our performance evaluations which compare PLAN-B, which are then presented in Section 7.5. Finally, Section 7.6 wraps up with a conclusion and an outlook on future research opportunities.

7.2 Context-aware broadcasting in MANETs

A broadcast is defined as the dissemination of one message in the entire network. This operation can serve as building block for all sorts of communication abstractions from unicast to publish/subscribe and it is therefore important that it provides a high level of Quality of Service (QoS). In evaluating different broadcast protocols, we are interested in two main measures of QoS namely, *reliability* and *efficiency*.

- *Reliability* is measured by the *delivery ratio*, i.e., the number of nodes that deliver a message m that was previously broadcast compared to the number that of nodes that should have delivered the message. An protocol is said to be reliable if it ensures a delivery ratio equal to 1.
- *Efficiency* is measured by the *number of forwards*, i.e., the number of retransmissions of a message m generated by a single broadcast. Efficiency goes down as the number of forwarders goes up.

Based on these QoS measures, we can say that the problem of broadcasting is to maximize both reliability and efficiency.

7.2.1 Broadcasting protocols

Intuitively, a broadcast protocol allows to disseminate a message m in the network so that all nodes in the network deliver m. In order to do so, broadcasting protocols offers the following two primitives:

- LFS-BROADCAST(m): broadcasts m to all nodes.
- DELIVER(m): works as a callback when m is received.

The simple flooding protocol solves the broadcast problem in a straightforward way: when a message is received for the first time, it is forwarded. However its inefficiency can lead to a broadcast storm [15]. Other protocols go further and try to solve the problem while meeting the extra challenge of minimizing the number of nodes that retransmit a message. A simple solution that offers increased efficiency is the simple GOSSIP1 protocol [15]. In this protocol, when a message is received, a node only forwards it with a probability p. Thus, the efficiency of gossiping can be fine-tuned by varying p. Unfortunately, the reliability of the protocol declines rapidly as p decreases. A promising approach to increase efficiency and reliability is to depart from context-oblivious approaches and to consider context-aware protocols.

7.2.2 Context-aware broadcast protocols

A protocol is said to be *context-aware* if it can collect information about its environment at runtime and act accordingly. Such information can include among others: location, network traffic, or signal strength. Context-aware broadcast protocols typically use a *wait and count* mechanism where they set a waiting time before retransmitting a message and cancel their scheduled retransmission if they hear some of their neighbors retransmitting the message before their own retransmission is due [3, 4, 5, 7, 14, 1, 16]. The idea of such protocols is that message retransmission is delegated to nodes that scheduled the shortest waiting time. This allows to increase efficiency without jeopardizing reliability or timeliness.

7.2.2.1 Setting schedules

Different approaches can be considered in order to set the waiting time. Some, such as CBS set the time randomly [5]. However, it would be more useful if nodes located at the outskirts of a sender's neighborhood forward messages instead of nodes located close to the sender, since the former can reach a greater uncovered area than the latter. In order to achieve this, protocols such as PAMPA set the time according to the proximity with the sender [3, 13], the closer the sender the longer the waiting time. In PAMPA, the distance between the sender and the receiver is computed based on the intensity of the signal with which the message is received. This approach is called power-aware. Other approaches, such as 6SB [4] or OFP [16] set the time according to the proximity with an optimal geographical locations. In the *six-shot broadcast* (6SB) protocol for instance, message forwarding is delegated to nodes located nearby six strategic positions. These positions, called *targets* are evenly spread on the edge of the sender's neighborhood. The idea is to compute a waiting delay when a message is received according to one's proximity to the closest target, the closer the target the shorter the waiting delay. In order to compute the targets, the location of the sender or the forwarder is embedded in the message.

7.2.2.2 Canceling schedules

Most of the previous protocols cancel their scheduled retransmission when they hear a certain number of neighbors retransmissions. Typically, the number of retransmission needed to cancel one's schedule is set by a threshold k. For instance, this is the strategy used by 6SB, PAMPA, OFP, and CBS. Other protocols fine-tune the cancellation process by attributing different weights to different retransmission based on their context [7, 3, 1].

7.2.3 Adaptive protocols

The protocols presented so far offer a wide range of solutions that can prove to be optimal in certain settings. However, if the setting change dynamically, these protocols can turn out to be suboptimal due to their hardwired parameters. In order to overcome this shortcoming, some authors proposed an adaptation mechanism to fine-tune protocol parameters at runtime. For example, SMARTGOSSIP [11] and HYPERGOSSIP [8] are two protocols that adjust the retransmission probability of a simple gossip protocol. SmartGossip, evaluates the importance of each node in the network and adjusts its retransmission probability accordingly. HYPERGOSSIP allows to fine-tune the GOSSIP1 protocol according to network density and aims at overcoming network partitions by implementing a rebroadcast mechanism. These works only allow to fine-tune the GOSSIP1 protocol, which offers only limited performance. Another example is the @FLOOD [14] protocol that continuously evaluates different protocols in the background and switches when a more efficient one is detected. Typically, @FLOOD evaluates 6SB, PAMPA and GOSSIP1.

We argue that there is a need for a lightweight adaptive context-aware broadcasting protocol that builds on existing state of the art mechanisms in order to provide enhanced performance in changing settings. Hereafter we present PLAN-B as a response to this need.

7.3 PLAN-B

PLAN-B is a novel proximity-based lightweight adaptive broadcast protocol that allows to fine-tune its forwarding decisions using information about the network density in order to ensure optimal reliability and efficiency in every setting. In order to do so, PLAN-B provides policies, which determine the adequate parameters to use depending on the number of nodes expected or detected to be present in the network. Formally, a policy can be expressed a a function $f(d) \rightarrow (z, k_p, k_s)$ that returns a set of parameters (z, k_p, k_s) for a given density d. We will discuss these parameters in further details shortly. It should also be noted that PLAN-B builds upon a wait and count mechanism, where relative distance between nodes is used to set schedules, therefore it can use either location sensors, power sensors, time sensor or a combination of the three in order to evaluate distances. This makes it less vulnerable to lack of precision or sensor malfunction.

7.3.1 Architecture

In terms of the OSI model, PLAN-B is part of the *network layer* as illustrated in Figure 7.1.

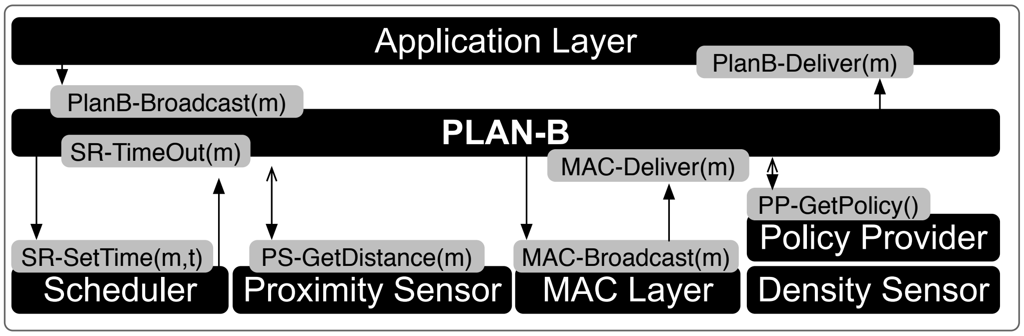


Fig. 7.1 Protocol Stack

Our communication model is based on a typical MANET, where nodes can emit messages that can be heard by all nodes in their direct neighborhood. PLAN-B is accessible to the application layer through the previously presented BROADCAST and DELIVER primitives, henceforth called PLANB-BROADCAST and PLANB-DELIVER to avoid confusion with their MAC layer counterparts. The MAC layer is a data link layer used to communicate with neighboring nodes. The MAC layer is just above the physical link between nodes. The MAC layer provides the following primitives:

- MAC-BROADCAST(m): broadcasts m to every node in the sender's transmission range.

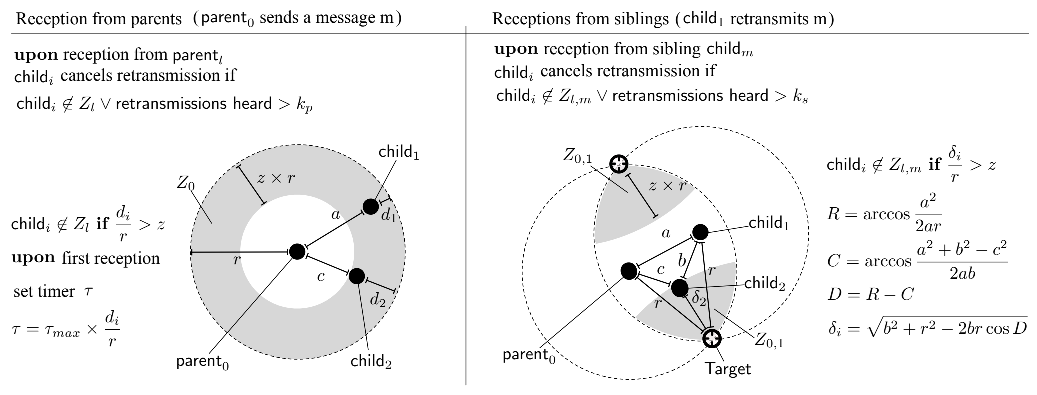

Reception from parents (parent$_0$ sends a message m)
upon reception from parent$_l$
child$_i$ cancels retransmission if
child$_i \notin Z_l \vee$ retransmissions heard $> k_p$
child$_i \notin Z_l$ if $\frac{d_i}{r} > z$
upon first reception
set timer τ
$\tau = \tau_{max} \times \frac{d_i}{r}$
Z_0
$z \times r$
child$_1$
a
d_1
r
c
d_2
parent$_0$
child$_2$
Receptions from siblings (child$_1$ retransmits m)
upon reception from sibling child$_m$
child$_i$ cancels retransmission if
child$_i \notin Z_{l,m} \vee$ retransmissions heard $> k_s$
$Z_{0,1}$
$z \times r$
child$_1$
a
b
c
r
δ_2
child$_2$
r
$Z_{0,1}$
parent$_0$
Target
child$_i \notin Z_{l,m}$ if $\frac{\delta_i}{r} > z$
$R = \arccos \frac{a^2}{2ar}$
$C = \arccos \frac{a^2 + b^2 - c^2}{2ab}$
$D = R - C$
$\delta_i = \sqrt{b^2 + r^2 - 2br \cos D}$

Fig. 7.2 PLAN-B principles for deliveries from parents and siblings. In the example, parent$_0$ is the first sender and child$_1$ the second.

- MAC-DELIVER(m): works as a callback when m is received.

A central component of the PLAN-B architecture is the proximity sensor (PS). Its specification allows a receiver of a message to retrieve the distance that separates him from the sender of the message. The implementation of PS can use various mechanisms, such as location sensor, power sensors or time sensors. For each of these implementations, messages must be customized to incorporate adequate information. This layer offers the following primitive:

- PS-GETDISTANCE(m): returns an estimate of the distance between the localhost and the sender of m.

The wait and count mechanism implemented in PLAN-B uses a scheduler (SR) with the following primitives:

- SR-SETTIME(m, t): schedules a waiting time t for m. The schedule is cancelled if $t = -1$.
- SR-TIMEOUT(m): works as a callback when the waiting time t for m has elapsed.

Finally, PLAN-B relies on a policy provider (PP). This module returns policies that contain specific values of parameters of PLAN-B based on current network densities returned by a density sensor. It offers the following primitive:

- PP-GETPOLICY(): gets the current value of the *policy*. As we will see, such a policy contains the threshold for the number of received messages from parents[1] (k_p) from siblings (k_s) and for the forward zone (z). Each of these variables can be access by dereferencing (e.g. *policy.k_p*).

[1] A parent of a node n with respect to a message m is a node that has not received its first copy of m from the same source as n has received its first copy of m. A sibling of n for m is a node that has received its first copy of m from the same source as n.

7.3.2 Principle

PLAN-B builds upon concepts from PAMPA and 6SB, i.e., delegating the forwarding task to nodes located at strategic points on the outskirts of the sender's transmission range. However, conversely to both 6SB and PAMPA, PLAN-B offers dynamic adaptation of its parameters through policies.

PLAN-B also builds on a wait and count mechanism with different behavior depending on the number of previously received retransmissions of a message. As we have seen, this means that there are two important tasks that the protocol does when receiving a message m. First, it must decide what schedule to set for a possible retransmission of m, if any. Second, it evaluates if a previously scheduled retransmission of m must be cancelled. PLAN-B differentiates between messages received from parents and form siblings. A message is said to be received from a sibling if both the sender and the receiver of the message had received a prior retransmission from the same node. A message is said to be received from a parent otherwise. The action that are undertaken when receiving a message form a sibling or a parent are described below and illustrated in Figure 7.2.

7.3.2.1 Receptions from parents

In Figure 7.2, parent_0 is the first sender. When she broadcasts message m nodes in her transmission range r, such as child_1 and child_2, receive it. Upon reception, child_1 and child_2 determine the distance d_i separating them from the outskirts of parent_0's transmission range. When the message is received for the first time, the children compute a waiting time proportional to the distance d_i. The message retransmission is cancelled for message recipients located outside the sender's forward zone Z_0 or if the number of redundant retransmissions of m is higher than the threshold k_p. If these conditions are not met before the waiting time elapses, m is retransmitted.

7.3.2.2 Receptions from siblings

When a node receives a retransmission of $m$from a sibling, it must evaluate wether or not to cancel the scheduled retransmission. This decision depends on the location of the receiver. Similarly to 6SB, PLAN-B computes locations, called targets, from where an additional retransmission would cover the most uncovered surface. Conversely to 6SB, these targets are not computed based on location, but only on proximity. Targets are located on the intersection between the neighborhood of the parent and the sibling as depicted in Figure 7.2. Receivers determine their distance δ_i to the closest target. The message retransmission is cancelled for message recipients located outside the target's forward zone $Z_{0,1}$ or if the number of redundant retransmissions of m is higher than the threshold k_s. If these conditions are not met before the waiting time elapses, m is retransmitted.

7.3.2.3 PLAN-B policies

As hinted above, forward policies determine the values for three parameters: the forwarding zone z, the threshold for the reception from parents k_p and the threshold for the reception from siblings k_s.

7.3.3 Implementation

Algorithm 7.14 details the implementation of PLAN-B. Upon initialization (lines 2-6), the transmission range r as well as the maximal waiting time τ_{max} are set. Furthermore, four lists are also initialized. The *parentCount* and the *siblingCount* list, used to keep track of the number of retransmissions received, the *parent* list, used to store the first parent to have sent a message m, and the *proximity* list that keeps a record of the distance between the receiver and the first parent.

7.3.3.1 Broadcasting

When PLANB-BROADCAST is executed, message m is broadcast on the MAC layer using the corresponding primitive (lines 7-8). Piggybacked on the message is the sender's ID and two empty field. These fields are reserved for retransmissions when the waiting time for a message elapses and the scheduler triggers the SR-TIMEOUT callback (lines 35-36). They provide information on the sender of the message (the parent) and its proximity to the current sender.

7.3.3.2 Delivering

When a message m is received, the MAC-DELIVER primitive is called. Using the proximity sensor, the *proximity* between the sender and the receiver of m is assessed and the waiting time for the message is initiated and it is determined whether the message is received from a parent or a sibling (lines 9-15).

For parent deliveries (lines 16-25), if it is the first delivery, m is delivered to the application through the PLANB-DELIVER callback, and The ID of the sender and it proximity are stored in the *parent* and *proximity* list respectively. Then a waiting time is scheduled, and the parent message counter incremented. Finally, it is evaluated whether or not to cancel the retransmission based on the number of retransmissions heard and the proximity of the last sender.

```
 1: uses MAC, Proximity Sensor (PS), Scheduler (SR), Policy Provider (PP)
 2: INIT
 3:     $r, \tau_{max} \leftarrow ...$                          {transmission range, maximal waiting time}
 4:     $parentCount, siblingCount \leftarrow \langle 0, 0, ..., 0\rangle$          {indexed by msg id}
 5:     $parent \leftarrow \langle \perp, \perp, ..., \perp\rangle$                  {parents, indexed by msg id}
 6:     $proximity \leftarrow \langle 0, 0, ..., 0\rangle$             {distances to parent, indexed by msg id}

 7: To execute planB-broadcast($m$) :
 8:     mac-broadcast($m$, getID(), $\perp$, $\perp$)                                  {broadcasts m}

 9: planB-deliver($m$) occurs as follows:
10:     upon  mac-deliver($m, srcID, parentID, a$) do                      {MAC delivers m}
11:         $prox \leftarrow$ ps-getDistance($m$)                             {src – receiver distance}
12:         if  $parentID \neq \perp$  and $parentID = parent_m$ then                  {tests kinship}
13:             siblingReception($m, a, prox, proximity_m$)                       {m from sibling}
14:         else
15:             parentReception($srcID, m, prox$)                                 {m from parent}

16: function parentReception($m, d$)
17:     if $parentCount_m = 0$ then                                  {test for first receptions}
18:         planB-deliver($m$)                                                  {delivers m}
19:         $parent_m \leftarrow srcID$                                        {assign parent for m}
20:         $proximity_m \leftarrow d$                              {assign parent proximity for m}
21:         sr-setTimer($m, \tau_{max} \times \frac{r-proximity}{r}$)                 {set schedule for m}
22:     $parentCount_m \leftarrow parentCount_m + 1$                         {increments m's counter}
23:     $policy \leftarrow$ pp-GetPolicy()
24:     if $r - d > policy.Z \times r$ or $parentCount_m > policy.K_p$ then
25:         sr-setTimer($m, -1$)                                        {cancels the schedule}

26: function siblingReception($m, a, b, c$)
27:     $R = \arccos \frac{a^2}{2ar}$                               {angle R parent-target-sibling}
28:     $C = \arccos \frac{a^2+b^2-c^2}{2ab}$                      {angle C parent-sibling-receiver}
29:     $D = R - C$                                          {angle D receveir-sibling-target}
30:     $\delta = \sqrt{b^2 + r^2 - 2br \cos D}$                      {receiver – target distance $\delta$}
31:     $siblingCount_m \leftarrow siblingCount_m + 1$                      {increments m's counter}
32:     $policy \leftarrow$ pp-GetPolicy()
33:     if $\delta > policy.Z \times r$ or $siblingCount_m > policy.K_s$ then
34:         sr-setTimer($m, -1$)                                        {cancels the schedule}

35: upon sr-timeOut($m$) do                         {when the waiting time for m has elapsed}
36:     mac-broadcast($m$, getID(), $parent_m, proximity_m$)                        {forwards m}
```

Algorithm 7.14 PLAN-B

For second deliveries (lines 26-34), the distance δ to the closest target is computed and it is evaluated whether or not to cancel the retransmission based on the number of retransmissions heard and the proximity of the target.

7.4 Performance evaluation settings

In the following, we describe the simulation environment and settings we use to perform our performance evaluation. We ran simulation on the *Sinalgo*[2] network simulator specifically aimed at communication protocols in wireless networks.

7.4.1 Communication scenario

In recent years, there was an increasing use autonomous robots in emergency situations for search, rescue or monitoring of hazardous products. Essentially, these robots are deployed to help rescuers after an earthquake or firefighters in a burning building [9, 2, 12]. In the aftermath of the March 2011 earthquake in Japan, the ongoing response effort at the crippled Fukushima Daichi nuclear power plant is the latest example where such robots are deployed [6]. In those situations, access to existing fixed infrastructure is not guaranteed [10], therefore decentralized communication is required. Typically robots have IEEE 802.11 capability to communicate with each other in order to coordinate their efforts. The communication scenario we use for our simulations is based on such a scenario, where nodes move across an open surface, equipped with location sensors and decentralized communication capabilities.

7.4.2 Density

We define three parameters to characterize the density and the degree of connectivity of the network: the *transmission range*, the *map size* and the *number of nodes*. All nodes have a transmission range between 80 and 100 meters using the Quasi Unit Disk Graph connectivity model (QUDG).[3] The map is a square of 1000 meters, i.e., 100 hectares,[4] or 10 hops across. To vary the node density, we vary the number of nodes in the fixed square map from 125 to 400 (1.25 to 4 nodes/ha). There is one sender per simulation.

[2] http://dcg.ethz.ch/projects/sinalgo/

[3] Nodes closer than 80 meters are assured to receive messages and nodes between 80 and 100 meters have an 80% probability of receiving messages.

[4] A hectare, abbreviated *ha*, represents a 100 meter-wide square.

7.4.3 Mobility

In our evaluation, we use a *torus random waypoint* (T-RWP) mobility model derived from the commonly used *random waypoint* (RWP) model. T-RWP overcomes the issue of node convergence to the center that can be observed in RWP by deploying nodes in a torus shaped map, which also avoids bound effects. We set the robot motion speed of the T-RWP model uniformly distributed between 1 and 2 meters per second (between 3.6 and 7.2 km/h). When a node reaches its waypoint, it waits for a uniformly distributed random delay, between 0 and 10 seconds. This accounts for the fact that nodes are not always moving. The message transmission time represents the delay between a LFS-BROADCAST call on one peer and the DELIVER callback on one of its neighbors.

7.4.4 Protocol specific parameters

Each of the previously presented protocols has its own specific parameters that determine its behavior. The p parameter for HYPERGOSSIP (between 0.2 and 1) represents the probability to forward a message when it is received for the first time. The *threshold* k is used in protocols that adopt the wait and count mechanism and is defined as follows: if a node receives a number of retransmissions (for a given message) that exceeds the threshold, the node will not forward that message. We test both a threshold of 1 and 2 for all protocols. As described previously, PLAN-B uses a a threshold for parent receptions (k_p) and a second threshold sibling receptions (k_s). We varied the former between 1 and 2 and the later between 0 and 2. The *max delay* set the upper limit for the waiting time before retransmitting a message. 6SB and PLAN-Bhave a parameter z expressed as the ratio of the transmission range and is used to compute the *forward zone* in which nodes may retransmit messages. With 6SB this parameter is set to 0.5 and with PLAN-Bit is set to 0.5 and 1.

7.5 Simulation results

Hereafter, we present the results of our performance evaluation in terms of reliability (delivery ratio) and efficiency (number of forwards). Each data point represents the mean of 100 simulations. We present these results in three steps. First, we detail the results of different static versions of PLAN-B in order to establish an optimal adaptive policy for different densities. Second, we compare PLAN-B to three static protocols, namely CBS, 6SB and PAMPA. Third, we compare PLAN-B to the HYPERGOSSIP adaptive protocol.

7.5.1 Establishing the adaptive policy for PLAN-B

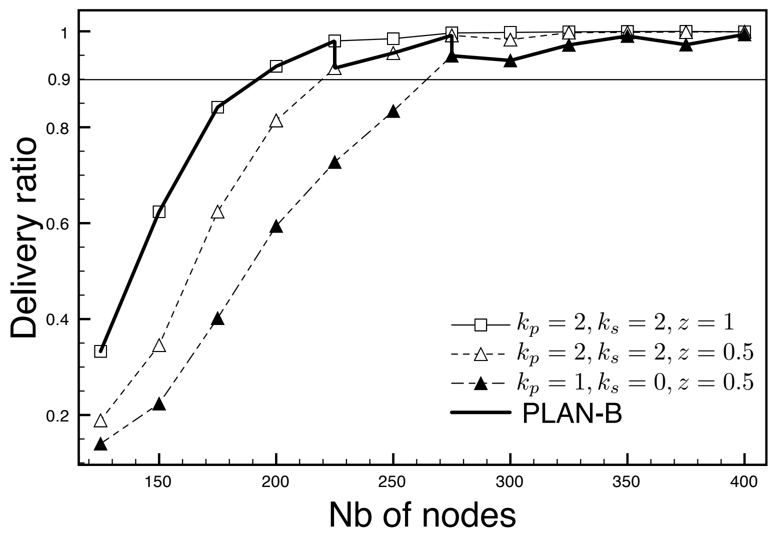


Fig. 7.3 Establishing an adaptive policy for PLAN-B.

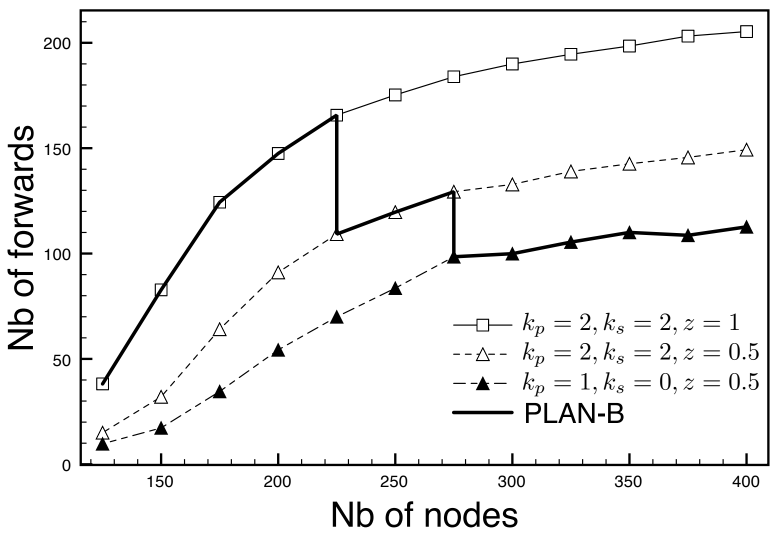


Fig. 7.4 Establishing an adaptive policy for PLAN-B.

We ran simulations of PLAN-B using the settings of parameters k_p, k_s and z described in the previous section. From these results, we selected three optimal settings for different densities, in order to establish the adaptive policy for PLAN-B as depicted in Figures 7.3 and 7.4. For each density, we selected a the most efficient setting that ensured at least a 90% delivery ratio (Figure 7.3).

Our results show that when less than 200 nodes are present in the network, the density is too low for the protocol to guarantee a 90% delivery ratio. When 200 nodes are present, our policy is to set the parameters to $k_p = 2,\ k_s = 2,\ z = 1$, which offers a 90% delivery ratio for a cost of up to 170 forwards. When over 275 nodes are present, our policy is to set parameters to $k_p = 1,\ k_s = 0,\ z = 0.5$. This offers the same delivery ratio for a cost of only 100-115 forwards. For densities in between, our policy is to set parameters to $k_p = 2,\ k_s = 2,\ z = 0.5$. This allows to reduce the message load by 30% compared to the low density alternative and still offer a 90% delivery ratio compared to the 70-80% delivery ratio offered by the high density alternative. Table 7.1 summarizes our policy.

Node number (N)	Nodes/ha (ρ)	Settings		
		k_p	k_s	z
$N < 225$	$\rho < 2.25$	2	2	1
$225 < N < 275$	$2.25 < \rho < 2.75$	2	2	0.5
$N > 275$	$\rho > 2.75$	1	0	0.5

Table 7.1 PLAN-B policy for different densities

7.5.2 Comparing PLAN-B to static protocols

We compare PLAN-B to three static context-aware protocols presented in Section 7.2, namely CBS, 6SB and PAMPA. These protocols rely on a wait and count mechanism with a threshold k. The results presented in Figures 7.5 to 7.10 show that this threshold can be set to $k = 1$ in dense settings for all protocols, but in order to ensure a 90% delivery ratio for settings with less than 250-300 nodes, a threshold $k = 2$ is required.

As these protocols cannot dynamically change these settings, a threshold $k = 2$ must be adopted in unknown or changing densities. In terms of reliability, PLAN-B and the other protocols ($k = 2$) reach a 90% delivery ratio in similar densities (200 nodes). 6SB reaches it a somewhat later (225 nodes). In terms of efficiency PLAN-B generates around 17% more forwards for settings with less than 225 nodes, but then allows to save up to 35% in medium and high densities. It should be noted that in high densities, PLAN-B even slightly outperforms CBS, 6SB and PAMPA with a threshold $k = 1$. So PLAN-B is a good choice in unknown or changing settings.

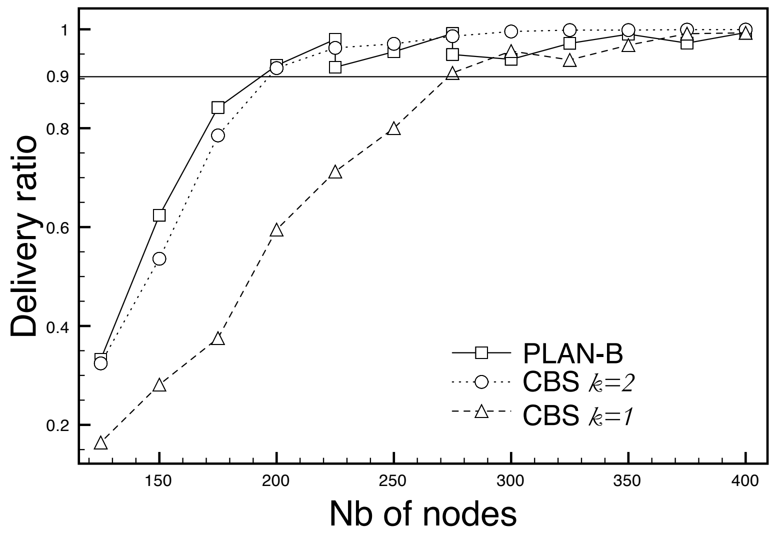


Fig. 7.5 Comparing PLAN-B to CBS.

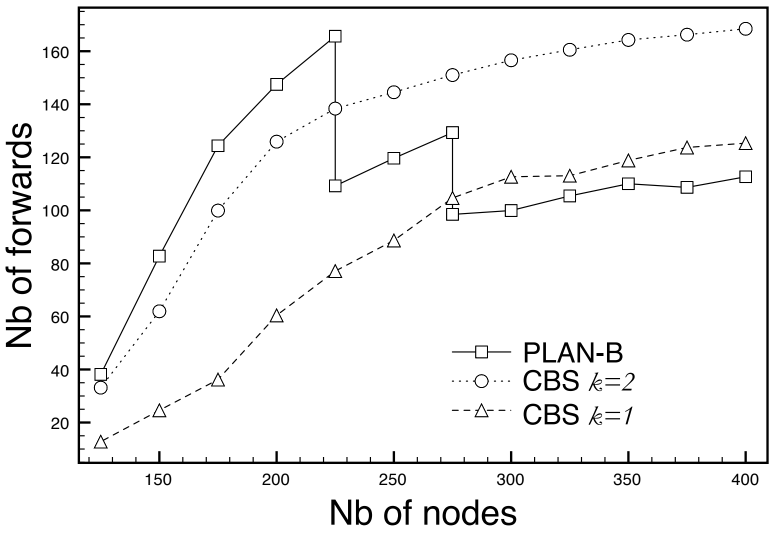


Fig. 7.6 Comparing PLAN-B to CBS.

7.5.3 Comparing PLAN-B to an adaptive protocol

Here, we compare PLAN-B to HYPERGOSSIP, an adaptive protocol that also adapts its parameters according to network density. More precisely, it uses this information to modify the forwarding probability p of GOSSIP1.[5]

[5] Our version of HYPERGOSSIP does not implement its mechanism to overcome network partitions, as we consider to be within one partition.

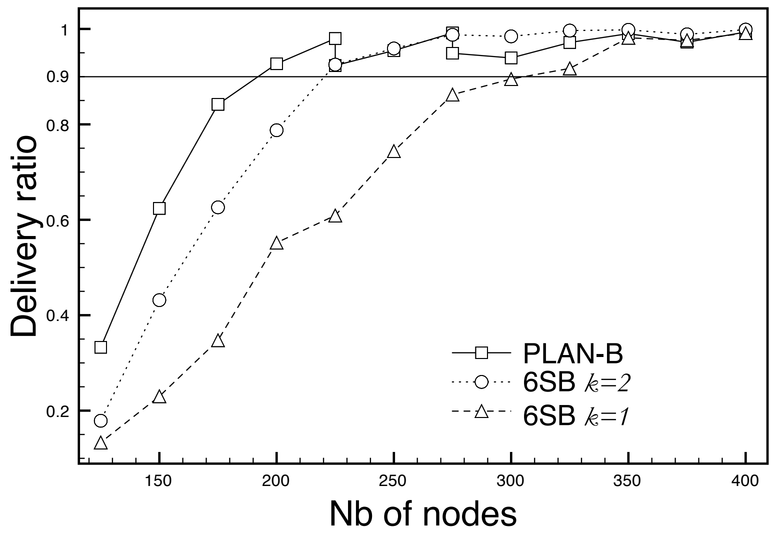


Fig. 7.7 Comparing PLAN-B to 6SB.

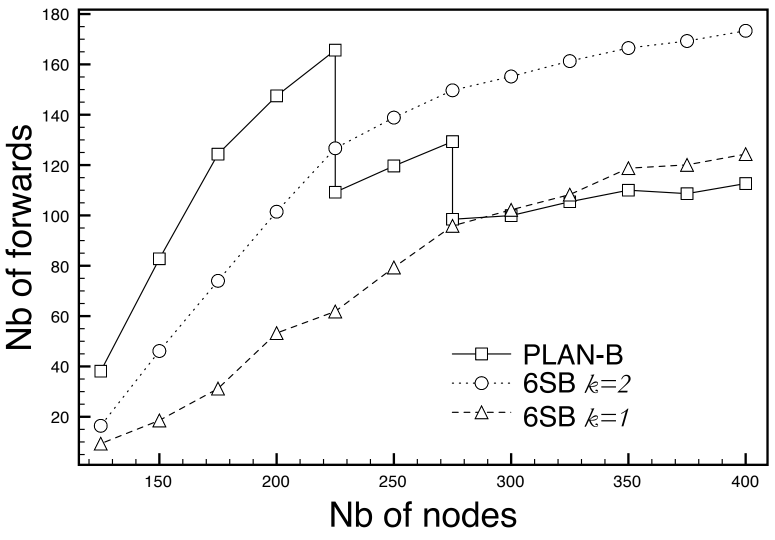


Fig. 7.8 Comparing PLAN-B to 6SB.

In order to choose the optimal value for p, we ran simulations for different a range of values of p from 0.1-1 and selected the lowest value of p for each density that offered a 90% delivery ratio. We compare these optimal values of p to PLAN-B in Figures 7.11 and 7.12. Our results show that HYPERGOSSIP does not reach a 90% delivery ratio before PLAN-B, making them similar in terms of reliability (Figure 7.11). In terms of efficiency, however, PLAN-B outperforms HYPERGOSSIP for any density (Figure 7.12). In dense settings, even though values of p continuously

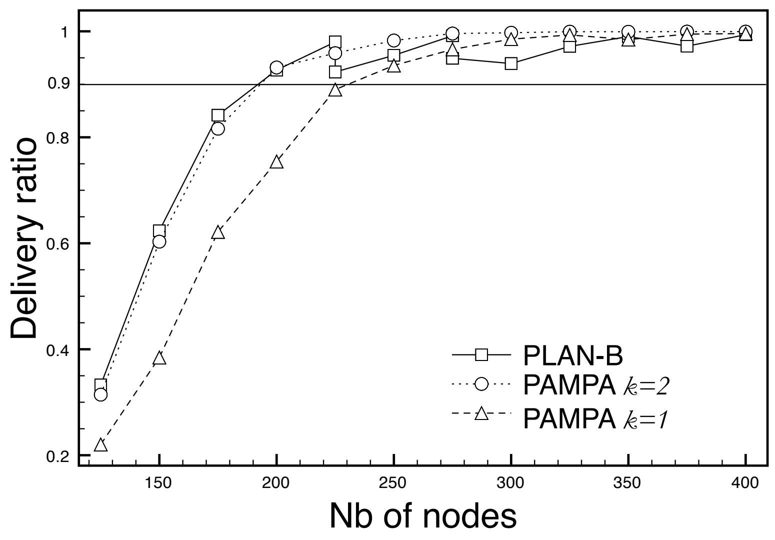


Fig. 7.9 Comparing PLAN-B to PAMPA.

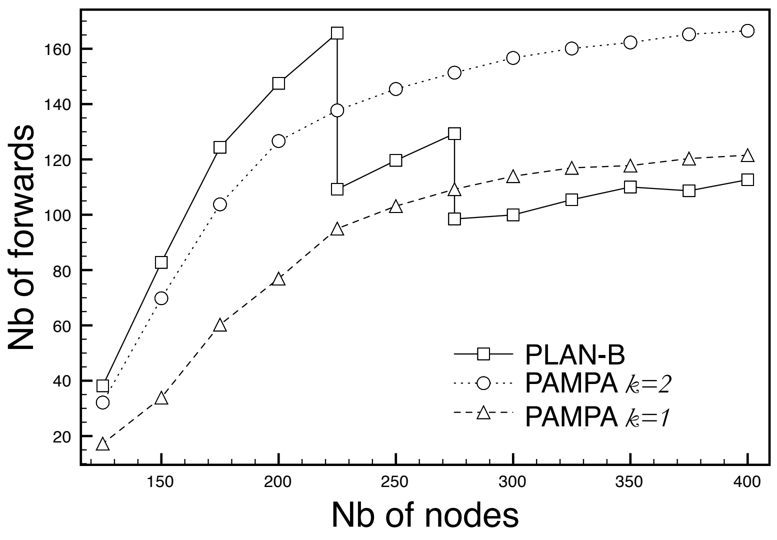


Fig. 7.10 Comparing PLAN-B to PAMPA.

decrease, PLAN-B reduces the number of forwards by a factor 2×, making it a very attractive alternative for any settings, as long as some proximity sensor is available.

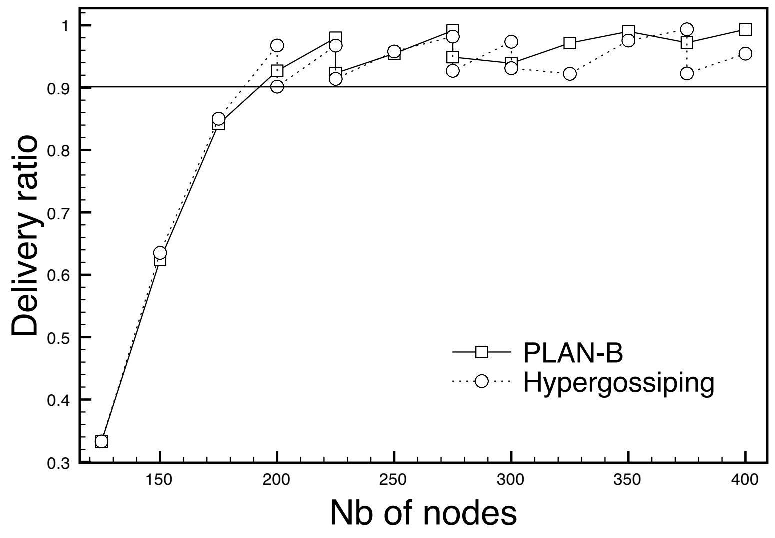


Fig. 7.11 Comparing PLAN-B and HYPERGOSSIP

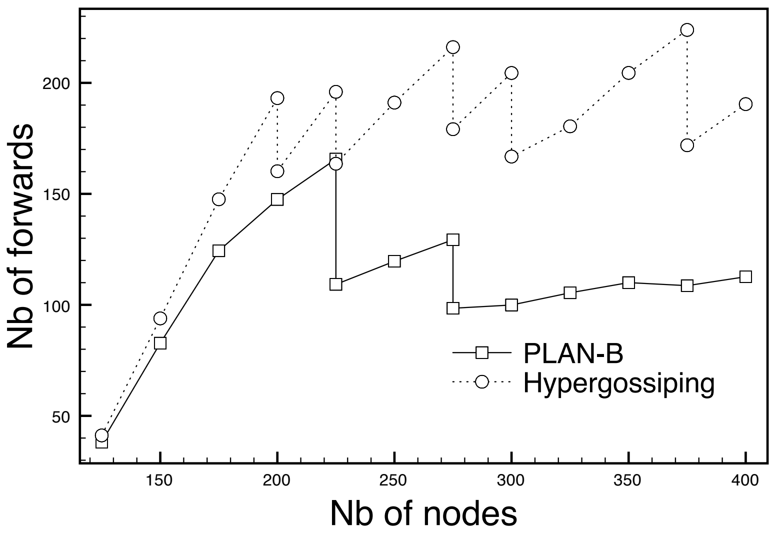


Fig. 7.12 Comparing PLAN-B and HYPERGOSSIP

7.6 Conclusion

This paper presented PLAN-B, a proximity-based lightweight broadcasting protocol for decentralized wireless networks, which can dynamically change its forwarding parameters depending on the network density. We compared it to several existing static and adaptive protocols and our performance evaluations show that, compared to existing methods PLAN-B offers increased efficiency for unknown, variable and dense settings. We are currently working on different implementations of our density sensor and we are investigating alternatives to furhter improve PLAN-B.

References

[1] Area-based beaconless reliable broadcasting in sensor networks. *IJSNet*, 1(1/2):20–33, 2006.

[2] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme. Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In *in IEEE International Conference on Robotics and Automation*, pages 3602–3609, 2004.

[3] C Ellis, H Miranda, and F Taïani. Count on me: lightweight ad-hoc broadcasting in heterogeneous topologies. *Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing*, pages 1–6, 2009.

[4] Benoît Garbinato, Adrian Holzer, and François Vessaz. Six-shot broadcast: A context-aware algorithm for efficient message diffusion in manets. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008*, volume 5331 of *LNCS*, pages 625–638. Springer, 2008.

[5] Z.J. Haas, J.Y. Halpern, and Li Li. Gossip-based ad hoc routing. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3:1707–1716 vol.3, 2002.

[6] T. Hornyak. Machines clear rubble as japan ranks crisis with chernobyl.

[7] Qing Huang, Yong Bai, and Lan Chen. Efficient lightweight broadcasting protocols for multi-hop ad hoc networks. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium*, pages 1 – 5, 2006.

[8] Abdelmajid Khelil, Pedro José Marrón, Christian Becker, and Kurt Rothermel. Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. *Ad Hoc Netw.*, 5:531–546, July 2007.

[9] Albert Ko and Henry Y. K. Lau. Robot assisted emergency search and rescue system with a wireless sensor network. *Intl. Journal of Advanced Science and Technology*, pages 69–78, 2009.

[10] Vijay Kumar, Daniela Rus, and Sanjiv Singh. Robot and sensor networks for first responders. *IEEE Pervasive Computing*, 3(4):24–33, October 2004.

[11] Pradeep Kyasanur, Romit Roy Choudhury, and Indranil Gupta. Smart gossip: An adaptive gossip-based broadcasting service for sensor networks. In *Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on*, pages 91 –100, 2006.

[12] F. Matsuno and S. Tadokoro. Rescue robots and systems in japan. In *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, pages 12 –20, aug. 2004.

[13] Hugo Miranda, Simone Leggio, Luis Rodrigues, and Kimmo Raatikainen. Removing probabilities to improve efficiency in broadcast algorithms. In *Proceedings of the 5th MiNEMA Workshop, Middleware for Network Eccentric and Mobile Applications*, pages 20–24, 2007.

[14] J Mocito, L Rodrigues, and H Miranda. @ flood: Auto-tunable flooding for wireless ad hoc networks. *Euro-Par 2010-Parallel Processing*, pages 478–489, 2010.

[15] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping 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, New York, NY, USA, 1999. ACM.

[16] Vamsi Paruchuri, Arjan Durresi, and Raj Jain. Optimized flooding protocol for ad hoc networks. *CoRR*, cs.NI/0311013, 2003. informal publication.

Part III
Testbed Framework

Chapter 8
Developing, Deploying and Evaluating Protocols with ManetLab

Published in the proceedings of
The International Conference on
NETworked sYStems (NETYS'13)
Marrakech, Morocco, May 2-4, 2013
Publisher: Springer

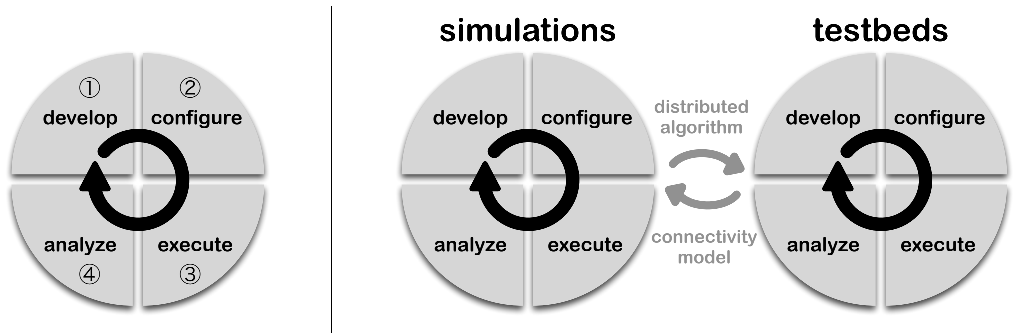
Abstract Evaluating the performance of MANET-specific communication protocols is essential to build robust mobile ad hoc applications. Unfortunately, most existing evaluation results are either based on simulations – which makes it difficult to draw conclusions beyond confined lab settings – or they are based on custom testbed results – which makes it difficult to reproduce them. In order to overcome this challenge, we introduce ManetLab, a modular and configurable software framework for creating and running testbeds to evaluate MANET-specific protocols. With ManetLab, one can easily configure and automate reproducible protocol executions on standard computer hardware, and thus provides both the *accuracy* of testbed-based evaluations and the *reproducibility* of simulation-based evaluations. After presenting ManetLab's extensible architecture, based on the notion of modular protocol stack, we show how it helps evaluate the performance of different broadcast protocols in real MANETs and how its results compare with simulation-based results.

8.1 Introduction

With the tidal wave created by the arrival of smart devices and tablets, the prospects of seeing MANET-based applications appear up in the distributed systems landscape has become more promising than ever. To encourage the emergence of such applications, system developers must provide solid communication building blocks for application developers, such as multi-hop broadcast, multicast, unicast, and other dissemination and routing protocols. Along that line, a large amount of research effort have been spent investigating mobile ad hoc routing protocols over the past decade. Central to this effort are the specialized tools that allow researcher to *develop* and *evaluate* their protocols.

8.1.1 Protocol development and evaluation

The development of an effective and efficient protocol, be it wired, wireless infrastructure-based or ad hoc, is an iterative process consisting of four steps, as illustrated in Figure 8.1a. For a start, one has to devise the protocol in the form of a distributed algorithm, ideally proving it formally and ultimately implementing it in some programming language (Step 1). Then, one has to configure some test environment in which the protocol will be executed (Step 2) and run the actual tests (Step 3). Finally, one has to analyze the collected data (Step 4), which might then lead to fine-tune the protocol and trigger a new iteration.



(a) general iterative process (b) combining simulations and testbeds

Fig. 8.1 Development of communication protocols

Existing tools for evaluating protocol performance can be categorised as either *simulators* or *testeds*. When using testbeds, the iterative process sketched in Figure 8.1a can be very time consuming, especially if one wants to evaluate performances in various distributed environments, which is clearly a must. This is particularly true for Steps 2 and 3, since they imply the deployment of the protocol code and of the test configuration to various distributed nodes, the launching of the testbed execution and the gathering of the results obtained at each individual node. For this reason, dedicated tools aimed at facilitating the creation and execution of testbeds have been proposed by the research community for specific families of distributed environments. This is for example the case of PlanetLab[1] for large-scale distributed systems [9].

Performance evaluation tools for MANETs. When it comes to evaluate the performance of MANET-specific protocols however, until now researchers have had essentially the choice between the reproducibility offered by simulators and the accuracy offered by testbeds. On the positive side, simulators are widely used by the research community and their source code is generally accessible online, which makes

[1] `http://www.planet-lab.org`

simulation-based evaluations fairly reproducible. Unfortunately, as they rely on the modeling of complex physical and logical parameters, it is difficult to draw general conclusions about the behavior of such protocols in real settings [26]. Section 8.5 further discusses evaluation tools for MANETs.

Testbeds on the contrary rely on real mobile ad hoc networking and therefore tend to offer a high-level of accuracy. For this very reason however, they also tend to impose a high development and deployment barrier [1]. In addition, most testbeds are not directly available to other researchers and often require specialized or incompletely specified hardware. For these reasons, the level of reproducibility of the resulting performance evaluation is generally quite low.

8.1.2 Contribution and roadmap

The ManetLab framework precisely aims at filling this gap, by supporting both accurate and reproducible performance evaluations of MANET-specific protocols, in a similar way PlanetLab does it for large-scale distributed systems.

In Section 8.2, we discuss the need for a tool such as ManetLab and its key requirements to achieve high accuracy and reproducibility. In Section 8.3, we present ManetLab in detail, by showing how it helps develop protocol layers and assemble them into a full protocol stack, which can then be deployed on remote nodes. We also introduce ManetLab's graphical tool, which helps configure performance evaluations, launch them and gather the corresponding results. In Section 8.4, we then compare the results of various performance evaluations obtained using ManetLab with those obtained with two simulations tools. While simulation-based performance evaluations are fairly accurate for simple MANET environments, they diverge significantly from the results obtained in reality for more complex environments, typically involving physical obstacles between nodes. Interestingly, when injecting the topological constraints observed with ManetLab back into the two considered simulators, simulation-based evaluation tend to augment their level of accuracy. Finally, we discuss existing testbeds for MANETs in Section 8.5, and ongoing work on improving ManetLab in Section 8.6.

8.2 Achieving accuracy and reproducibility

To be useful, performance evaluations of MANET-specific protocols should obviously achieve a high degree of both accuracy and reproducibility. As a consequence, tools supporting such evaluations should mimic real-life environments as accurately as possible and should allow researchers to easily reproduce experiments with the purpose of comparing evaluation results.

8.2.1 Combining simulations and testbeds

Although this paper focuses on the need for a robust testbed tool, we advocate the combination of simulations and testbeds, as illustrated in Figure 8.1b. While simulations can indeed be considered an acceptable first approximation, communication protocols tend to perform in a more unpredictable way when actually deployed in a real MANET than in an infrastructure-based network, be it wired or wireless. For this reason, we believe that ultimately accuracy can only be achieved with testbed approaches, at least at that stage of the evaluation.

In the early phases of a protocol development, simulations can be very useful to validate the protocol in simple environment settings, e.g., in the absence of walls or obstacles, using a basic wireless propagation model. Once this first validation is done, the protocol should then be evaluated in a real mobile ad hoc network, using testbeds. As shown in Figure 8.1b, the results of testbed-oriented evaluations can then be injected back into simulations, typically in the form of a more accurate model of the wireless connectivity among network nodes. This is precisely the approach we follow in Section 8.4.

8.2.2 Creating accurate and reproducible testbeds

While various simulation tools exists, some of which have become de facto standards, the situation is very different when it comes to testbeds for mobile ad hoc networks. Moreover, coming up with universal and rigid testbeds for MANETs might not even be a desirable goal, given the great variability of actual deployment settings. Rather, we believe that there exists a need for a framework that facilitates the development, deployment and evaluation of MANET-specific protocols, by making it easy to create accurate and reproducible testbeds. That is, *accuracy* and *reproducibility* should be the two key requirements for such a testbed framework.

Accuracy. Devising a tool that offers an accurate evaluation of the behavior of a protocol running in a MANET can be very challenging. The central issue stems from the fact that MANETs tend to exhibit very erratic behaviors in terms of connectivity and of reliability, depending on their physical environment and on how nodes are moving. In addition, the various layers that stack up, in particular TCP/IP, tend to distort the actual performance evaluation of communication protocols in MANETs. For this reason, an adequate testbed framework should offer flexibility in protocol layering, all the way down to the lowest-level layers, typically by making it easy to compose protocol stacks from elemental layers.

Reproducibility. To evaluate the accuracy of results provided by an evaluation tool, other researchers must be able to reproduce the testbeds described in the literature, and scrutinize the evaluation tool itself. Thus, it is important that any evaluation tool, in particular a testbed framework, be easily accessible for the research community. A testbed framework should in addition be configurable, in order to easily switch from one deployment setting to another, and it should offer support for automatically launching evaluations and gathering results.

8.3 Introducing ManetLab

ManetLab is a framework supporting the creation and execution of accurate and reproducible testbeds for MANETs-specific protocols, using mainstream hardware.[2] On each computer where ManetLab is installed, the wireless network interface is used to connect the MANET, while the wired network interface is used as control network to provide feedback about the protocol performance. More specifically, as illustrated in Figure 8.2, each computer running ManetLab is hosting an *agent* connected to the MANET. In addition, one of the computers hosts the *controller*, which acts as a conductor orchestrating the protocol execution.[3] That is, the controller uploads the protocol stack to each agent, triggers the execution of the protocol and collects feedback from all agents about the protocol execution, via the wired network interface. In the following, we discuss how each step pictured in Figure 8.1 is performed with ManetLab.

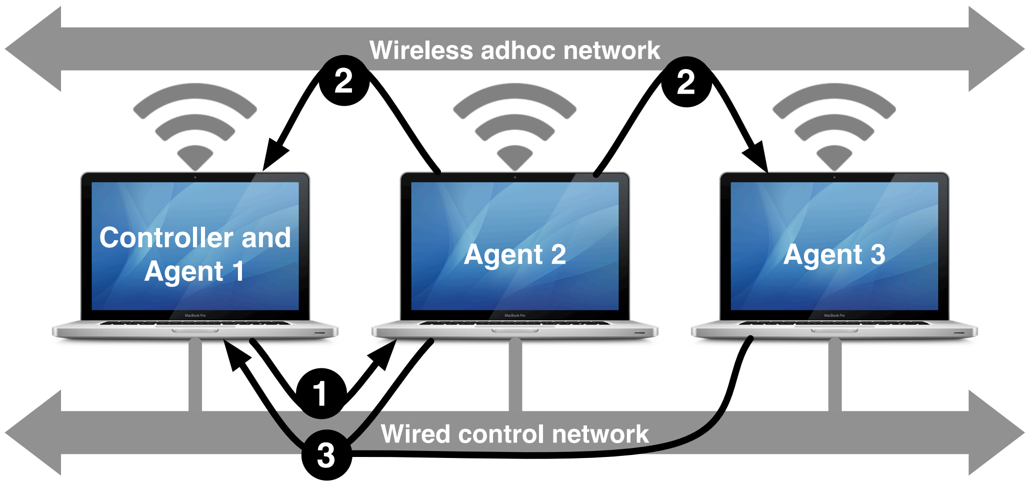


Fig. 8.2 ManetLab — Using an ad hoc network and a control network

[2] ManetLab runs on Apple's computers with Mac OS X 10.7 or higher.
[3] The example depicted in Figure 8.2 is further discussed in Section 8.3.3.

8.3.1 Development

In order to test an ad hoc protocol, one has to first implement it. ManetLab proposes an API[4] to help MANET-specific protocol developers in this task. The protocol must be implemented in Objective-C and designed as layers, inheriting from the MLStack-Layer class, in a stack (MLStack) provided by the API. The layer above the stack represents the application, whereas the layer below the stack is the antenna. At any given time, a ManetLab node is executing at most one protocol stack. Communication between nodes and between layers inside a stack is achieved via message passing (MLMessage). Figure 8.3 illustrates a stack containing two layers, i.e., a fragmentation layer and a gossip layer.

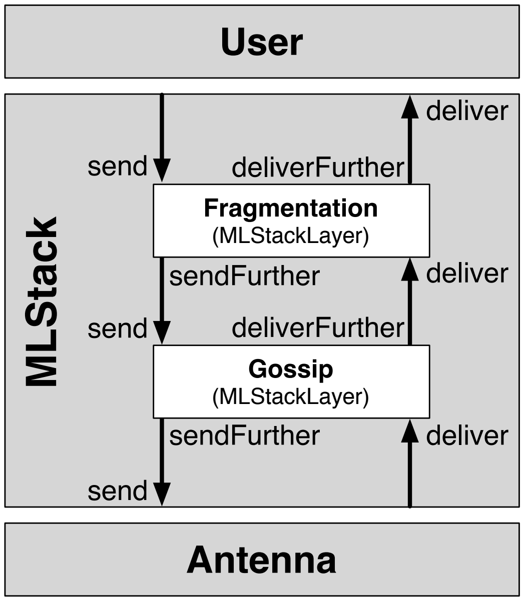
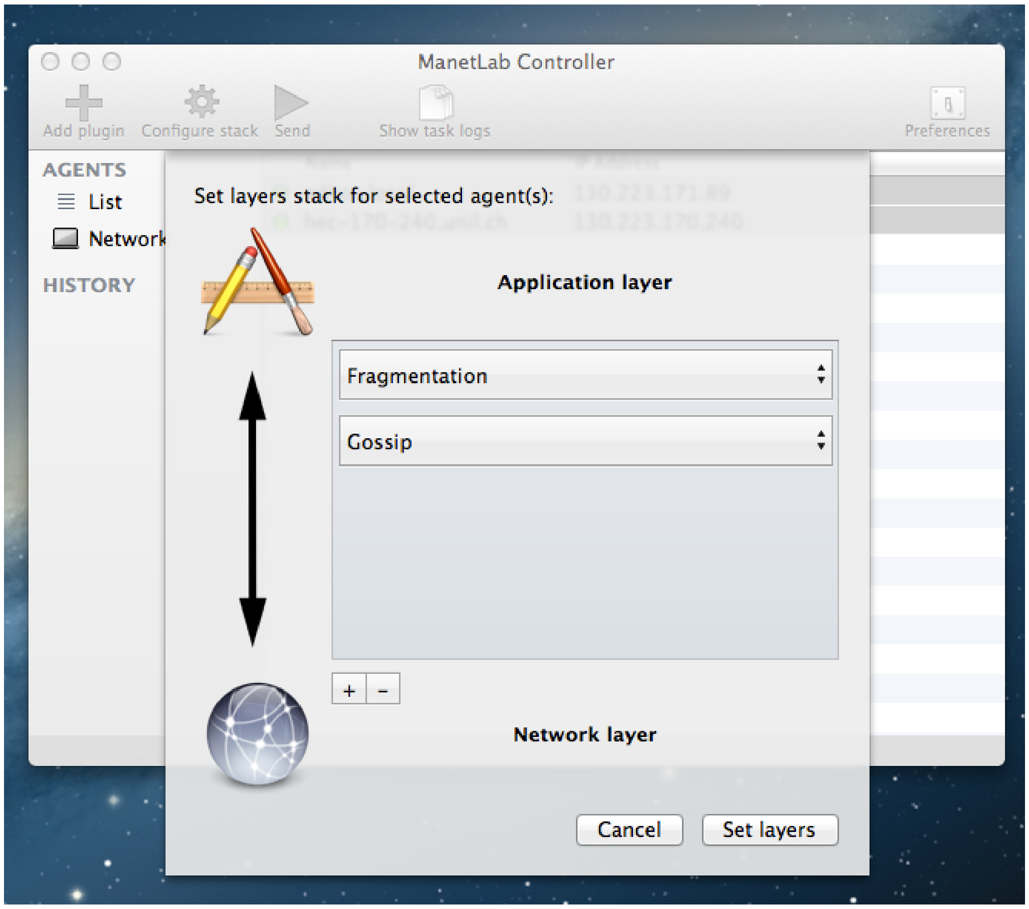


Fig. 8.3 A stack with two layers.

As pointed out in [26], researchers rarely make the effort to provide the source code of their MANET-specific protocols to the community, which makes it very difficult to seriously compare different protocols pursuing the same goal. Moreover, even when the source code is provided, the absence of a standardized tool to compose and deploy protocols leads to low reproducibility of most research results. For this reason, ManetLab proposes a plugin architecture that makes it easy to package all the layers (subclasses of MLStackLayer) and message types (subclasses of MLMessage) composing a MANET-specific protocol stack. As a result, stacks created using ManetLab can be shared and reused by other researchers. In addition, ManetLab is

[4] The API is distributed with the ManetLab software and available at `http://doplab.unil.ch/manetlab`



Set layers stack for selected agent(s):

Fragmentation

Gossip

Fig. 8.4 GUI corresponding to Figure 8.3.

an open source project, which further promotes the scrutiny and reproducibility of testbeds relying on it.

8.3.2 Configuration

The configuration step greatly differs in a simulator-based approach and in a testbed-based approach. However, in both cases, configuration entails *installation* of the tool (simulator or testbed) and its *parameterization*, and then *deployment* of the protocol code in the tool.

Installation. Installing a simulator is usually non-trivial because it implies to download the source code from the Internet and then to build the simulator from that code. As for testbeds, they are rarely made publicly available and when they are, they tend to be even more difficult to install and to use, since they often require specialized hardware. In contrast, ManetLab requires just a few clicks to be installed on a standard desktop or laptop computer.[5] Yet, if one wants to access its source code, it is also made available via GitHub.[6]

Parameterization. For both simulators and testbeds, parameterization implies to load the protocol stack into the tool, usually in some binary form. Apart from this obvious step, parameterization is where simulators and testbeds differ the most. As testbeds rely on real MANET implementations, one has only a few parameters to set, e.g., the wireless channel used to communicate; this is typically the case with ManetLab. In addition, as already suggested in Section 8.3.1, ManetLab makes it easy to dynamically load plugins containing protocol layers into the tool, to then graphically compose a protocol stack from these layers and deploy it of each node of the MANET (Figure 8.4). When it comes to simulators however, many more parameters have to be set, such as the mobility model, the connectivity model, the node distribution model, the interference model, etc. In terms of *accuracy*, this step is critical because unrealistic values may result in misleading or even erroneous performance evaluations.

Deployment. When using testbeds, one of the major obstacles to reproducibility often lies in the need to deploy and maintain specialized hardware, typically in the form of prototype devices. In order to solve this problem, at least partially, ManetLab is implemented on the OS X platform, which is widespread in research institutions today. In addition, Apple's hardware is known to be very standardized and traceable, e.g., using a tool like Mactraker,[7] which is clearly an advantage in term of *reproducibility*. Moreover, building ManetLab on top of OS X, which shares the same code basis as iOS when it comes to low-level services, opens the opportunity to port ManetLab to iOS devices in the future.

8.3.3 Execution

With ManetLab, protocols communicate using the IEEE 802.11 wireless ad hoc network (IBSS mode), so the accuracy of performance evaluations is naturally ensured. On the controller, the graphical user interface shown in Figure 8.5 is used to prepare and launch the testbed execution.

Execution example. Arrows pictured in Figure 8.2 illustrate an execution with ManetLab, where the controller simply requests one agent to broadcast a message. First, the controller asks Agent 2 to broadcast a message (Arrow 1). As a result,

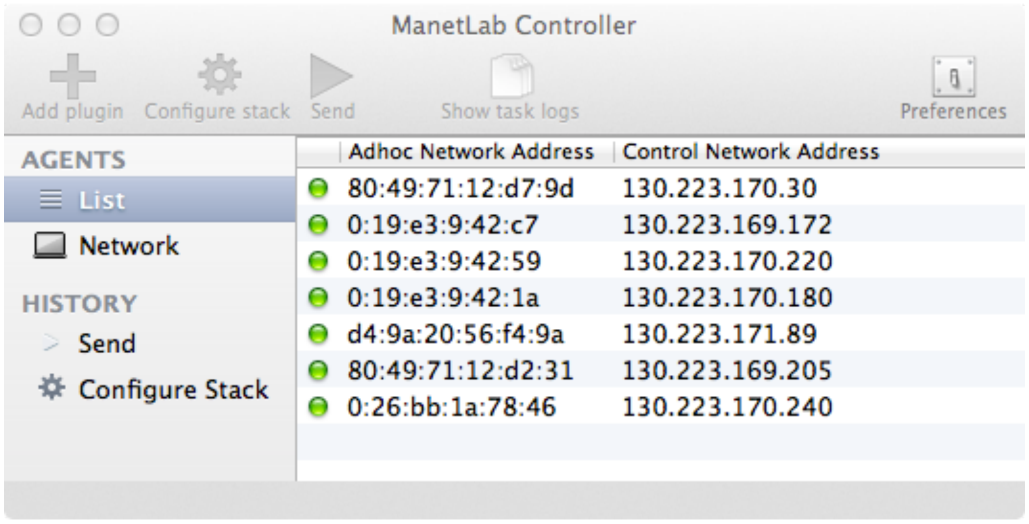
[5] ManetLab executable is available from `http://doplab.unil.ch/manetlab`.

[6] ManetLab source code is available from `http://github.com/doplab/ManetLab`.

[7] `http://mactracker.ca`

Agent 2 does indeed broadcast a message on the wireless ad hoc network (Arrow 2). Finally, all agents have received the message and provide feedback to the controller, using the wired and reliable control network (Arrow 3).

Offline control mode. Since the controller communicates with agents via a wired control network, ManetLab does not allow to test protocol with mobility in its first version. To overcome this limitation, we are currently implementing an *offline control mode*, which uses the wireless network for both control messages and protocol messages. The idea is to have each agent log its evaluation results locally during the testbed execution, so that it can send them to the controller after the execution.



Adhoc Network Address	Control Network Address
80:49:71:12:d7:9d	130.223.170.30
0:19:e3:9:42:c7	130.223.169.172
0:19:e3:9:42:59	130.223.170.220
0:19:e3:9:42:1a	130.223.170.180
d4:9a:20:56:f4:9a	130.223.171.89
80:49:71:12:d2:31	130.223.169.205
0:26:bb:1a:78:46	130.223.170.240

Fig. 8.5 ManetLab Controller — Graphical user interface

8.3.4 Analysis

Most simulators and testbeds do not provide specific analysis tools. Rather, they allow protocol developers to produce log files or populate databases, which can then be fed into some analysis tool, such as a graphical network animator like NetAnim for example. This is also the case of the ManetLab testbed framework: its API offers a log methods that allows developers to produce whatever trace they need for their performance evaluation.

8.4 ManetLab testbeds *versus* simulations

To compare performance evaluations obtained from ManetLab with those obtained from simulations, we study the behavior of various broadcast protocols in ManetLab and in two simulators. These two simulators are ***NS-3*** [15], the latest simulator from the *NS* family, and ***Sinalgo***,[8] a simple Java-based simulator we used to evaluate several of our own MANET-specific protocols [12, 13, 11]. For our comparison, we rely on two simple measures: the ***delivery ratio***, defined as the *number of nodes who received a message over the total number of nodes*, and the ***forward ratio***, defined as the *number of nodes who send or retransmit the message over the total number of nodes*. Each measure is the average of 1'000 distinct executions.

8.4.1 Network settings

In order to avoid a potential distortion or overhead caused by TCP/IP (in particular its routing scheme), all protocols are directly using the MAC layer when it comes to broadcast a message in the MANET. Along that line, we parameterize the MAC layer of each tool in a similar way, as discussed hereafter.

ManetLab. Since ManetLab is a real MANET implementation, there are only a very few settings we can change. All other settings are constraints deriving from the operating system and the hardware on which ManetLab is running. Basically, ManetLab creates an IEEE 802.11a ad hoc network with a theoretical data rate of maximum 6 MBits/s for broadcast.

NS-3. We use the *WifiNetDevice* from NS-3 with the *YansWifiChannel* and the *YansWifiPhy* models. We set all the settings we can to similar values of what ManetLab uses on real computers. That is, we configure NS-3 to use an IEEE 802.11a physical layer model, with a data rate of maximum 5.5 Mbits/s and a MTU of 1'500 bytes.

Sinalgo. Being a higher-level simulator than NS-3, Sinalgo does not rely on an implementation of the IEEE 802.11 standard. So we configure Sinalgo to send messages smaller than 1'500 bytes, with a *Unit Disk Graph* connectivity model and a *Signal to Interference plus Noise Ratio* interference model.

[8] `http://www.disco.ethz.ch/projects/sinalgo/`

8.4.2 Protocols, environments and communication patterns

Because we aim at providing a solid first comparison, we consider a number of *broadcast protocols*, *physical environments* and *communication patterns*. They are presented in details hereafter.

Broadcast protocols. We consider three probabilistic broadcast protocols, namely ***Simple Flooding***, ***Gossip***, and ***Counter-Based Scheme*** (CBS), which are well-known to the research community. With Simple Flooding [14], each node systematically retransmits a message the first time it receives it, so the delivery ratio is always equal to the forward ratio. With Gossip [29], each node retransmits a message with a probability p the first time it receives it. For our comparisons, we set p to 0.7, 0.5, and 0.2. Finally, with CBS [29], a node waits for some random delay between 0 and w_{max} before retransmitting a message, only if it received it only once. That is, if a node receives a message more than once, it does not retransmit it. For CBS, we set w_{max} to 0.1 and 0.5 seconds.

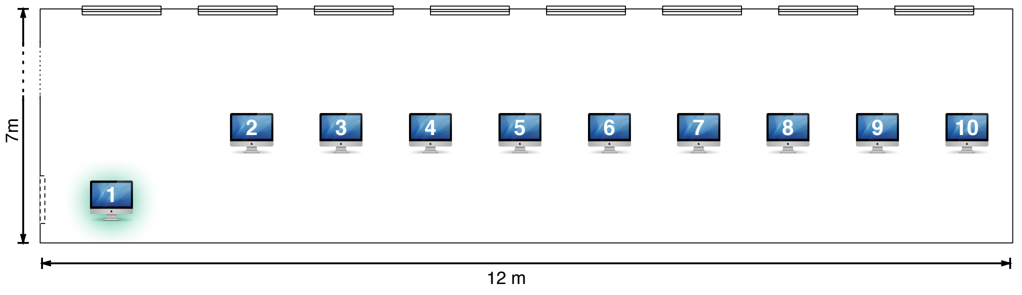


Fig. 8.6 Sketch of the open space environment.

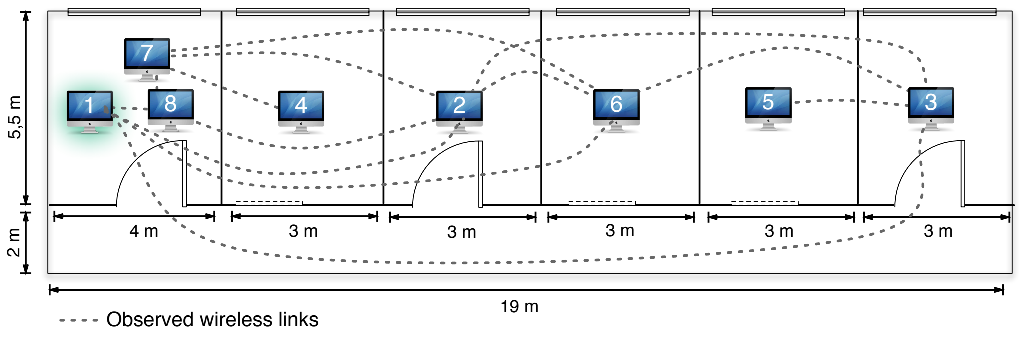


Fig. 8.7 Sketch of the private offices environment.

Physical environments. We consider two environments: an ***open space*** and ***private offices***. In the open space, ten computers are placed in one open space as illustrated in Figure 8.6; this is typically the case in a classroom. With private offices, eight computers are placed in adjacent offices as depicted in Figure 8.7. In each physical environment, Computer 1 acts as the initial broadcaster.

Communication patterns. We consider two communication patterns: a ***one-shot*** message, which corresponds to a low network load, and the ***streaming*** of 1'000 messages, which correspond to a high network load. With the *one-shot* pattern, Computer 1 broadcasts a single 1'400-bytes message. Those 1'400 bytes are encapsulated in just one network frame consisting of 1'485 bytes, including headers. With the *streaming* pattern, Computer 1 sends 1.4 Mbytes, which are fragmented into 1'000 network frames of 1'485 bytes each.

8.4.3 Results in the open space environment

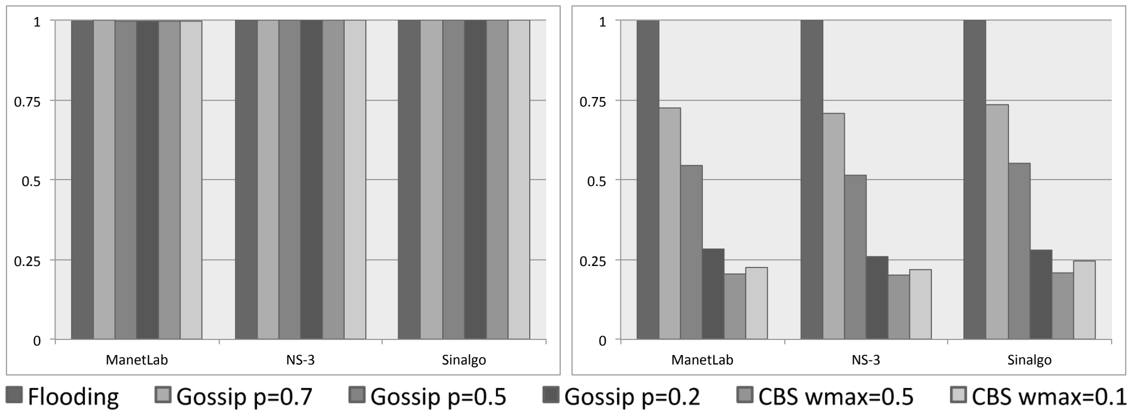


Fig. 8.8 One-shot in an open space – delivery ratio (left) and forward ratio (right).

Figure 8.8 shows the delivery and forward ratios of the one-shot communication pattern. Since all nodes are connected (fully connected graph) and there is almost no interference, the delivery ratios are strictly equal to 1.0 for all protocols in NS-3 and Sinalgo, and above 0.99 for ManetLab. For this reason, the forward ratios tend to converge towards their theoretical values, i.e., 1 for flooding, p for gossip and much smaller values for CBS. Overall, we can say that in this scenario (one-shot in an open space), simulations are quite accurate since they faithfully mimic the results obtained by ManetLab in a real MANET.

In the second scenario (streaming in an open space), interference starts to disturb the behavior of the protocols and affect both the delivery ratio and the forward

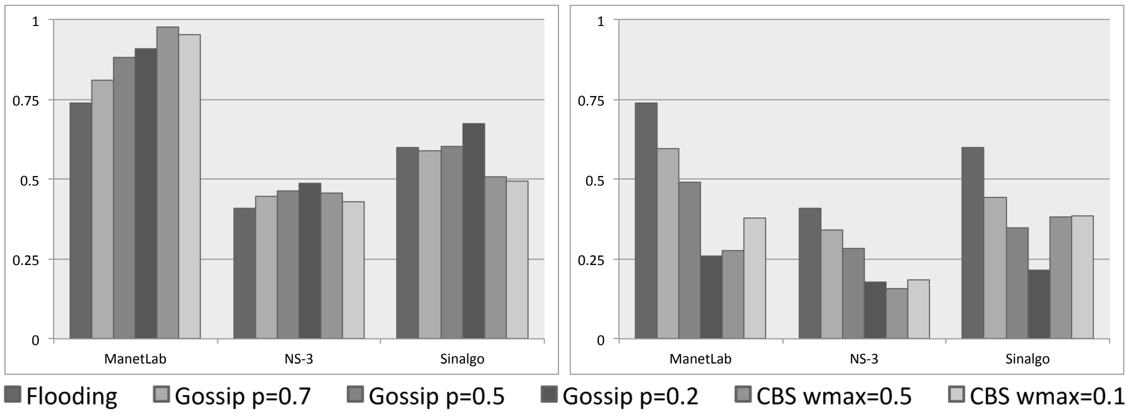


Fig. 8.9 Streaming in an open space – delivery ratio (left) and forward ratio (right).

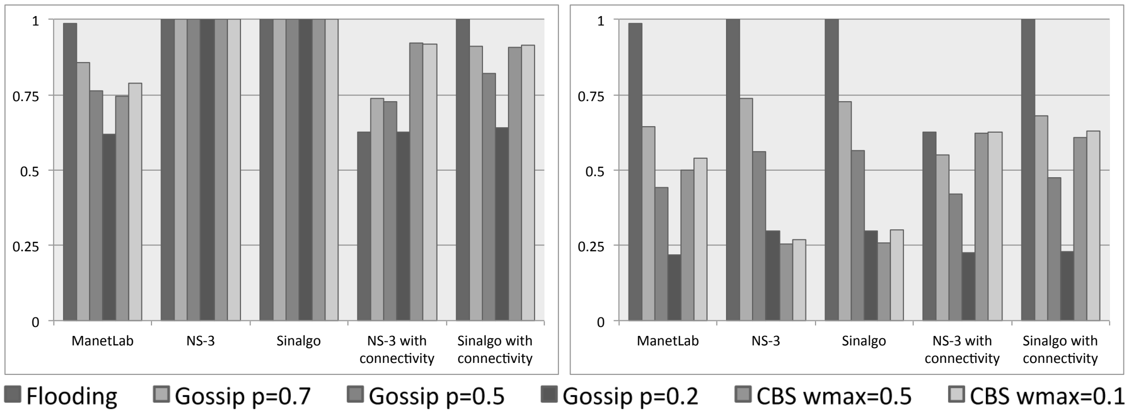


Fig. 8.10 One-shot in private offices – delivery ratio (left) and forward ratio (right).

ratio, as shown in Figure 8.9. The more messages are transmitted, e.g., for flooding or for gossip with $p = 0.7$, the more the delivery ratio decreases. Interestingly, the delivery ratios of ManetLab are over 0.7, whereas the delivery ratios of NS-3 and Sinalgo are under 0.6. That is, the interference models used in the two simulators are discarding too many frames, which indicates that their accuracy is diminishing. As for the forward ratios, they tend to be only slightly lower with the simulators than with ManetLab.

8.4.4 Results in the private offices environment

In the private offices environment, the real MANET experienced by ManetLab is no longer a fully connected graph, due to various physical obstacles (mainly walls but also furniture, possibly people, etc.). It is thus not surprising that the delivery ratio of a one-shot communication pattern in ManetLab tends to drop compared to the

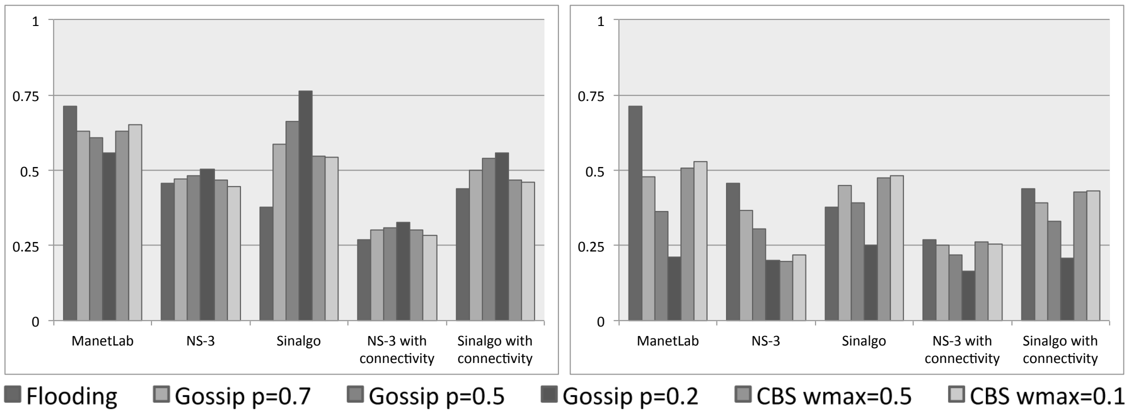


Fig. 8.11 Streaming in private offices – delivery ratio (left) and forward ratio(right).

open space, as shown in Figure 8.10. NS-3 and Sinalgo, on the contrary, continue to view the MANET as a fully connected graph, so their delivery ratios remain strictly equal to 1. This clearly indicates that their level of accuracy is dropping. As for the forward ratios, NS-3 and Sinalgo have similar results to those of ManetLab except for CBS.

Streaming in private offices is by far the worst scenario when it comes to the delivery ratio, as shown in Figure 8.11. Both physical obstacles and interference are significantly decreasing the performances of all protocols. Again, the accuracy of NS-3 and Sinalgo is compromised, as they only roughly approximate the delivery ratio observed with ManetLab. Furthermore, since fewer nodes receive the messages being broadcast, the forward ratios with NS-3 and Sinalgo are also dropping and thus diverge from those observed with ManetLab.

8.4.5 Injecting the observed connectivity into simulations

It seems reasonable to assume that the drop in accuracy we observe for both NS-3 and Sinalgo, when considering private offices, is largely due to their erroneous modeling of the MANET connectivity. In order to confirm this assumption, we inject the connectivity graph experienced by ManetLab (see Figure 8.7) into NS-3 and Sinalgo, and we re-run our performance evaluations.

As shown in Figure 8.10, after the injection the accuracy of both NS-3 and Sinalgo is improved for the one-shot communication pattern. Interestingly, and somewhat surprisingly, the results for the simple flooding protocol are more accurate with Sinalgo than with NS-3. With the streaming communication patterns however, injecting the connectivity graph is not sufficient to improve the accuracy of NS-3 and Sinalgo, as shown in Figure 8.11. It seems that the effect of interference, combined with a lower connectivity, leads both simulators to produce results that are significantly lower than what happens in reality.

8.5 Related work

To evaluate the behavior of their protocols, researchers should rely on simulators and testbeds that aim at providing *accurate* and *reproducible* performance evaluations. Hereafter, using these two dimensions, we review a wide range of evaluation tools for MANETs found in the literature [26, 28, 24, 16, 25] and we compare them with ManetLab. For accuracy, we focus on their communication support, as this is a critical element when it comes to evaluate performance in a MANET. For reproducibility, we assess the availability of the tools.

8.5.1 Communication support

Simulators on the one hand do not provide a real implementation of a wireless communication layer. For this reason, assessing the accuracy of their communication support, i.e., of their modelling of wireless communications, is very difficult and can only be achieved by comparing their results with those of a real MANET (as we did in Section 8.4). Such tools include NS-2 [8], NS-3 [15], GloMoSim [33] and its commercial version Qual Net[9], OPNET [7], OMNet++ [32], and others such as Sinalgo or JIST / SWANS [3, 4].

Testbeds on the other hand tend to be more accurate because they rely on real wireless communication links. Such tools include Castadiva [17], MASSIVE [27], MobiEmu [34], mLab [21], Carnegie Mellon University Wireless Emulator [22], ORBIT [30], Seawind [23] and WHYNET [35] or JEmu [18], PoEM [19], and of course ManetLab. Some testbeds however tend to oversimplify topological constraints, e.g., by simply piling up a stack of wireless devices. In addition, while one-hop communication is provided by all testbeds, multi-hop communication is only found in tools such as WHYNET, RoofNet [6], ManetLab and Airplug-emu [5]. Other testbeds also provide multi-hop communication, but they shorten their wifi range. Such tools include ORBIT, TrueMobile [20] and MiNT [10]. Other tools provide a logical multi hop communication, such as mLab, MobiEmu and Castadiva.

[9] `http://www.scalable-networks.com/content/products/qualnet`

8.5.2 Tool availability

To assess the availability of each tool, we evaluate if it is available *online*, if its *source code* is disclosed and available for download and installation, if detailed *documentation* is provided, and if *specialized hardware* is required in order to run testbeds relying on that tool.

Online availability. While many of the surveyed tools are available online for download, just as ManetLab, some other tools are only described in scientific papers, with no further details provided online. This is for instance the case of TrueMobile, PoEM, MASSIVE, JEmu, and the tool described by Barolli et al. [2]. This makes it very hard for other researchers to get a hold of these tools and reproduce experiments.

Source code availability and documentation. In order to evaluate the accuracy of a performance evaluation tool, providing the disclosed source code is another important aspect. Most reviewed simulators, except GloMoSim and OPNet, provide a downloadable version of their code. Among testbeds however, source code becomes much more scarce: only Castadiva, MobiEmu, mLab, Airplug-emu and MIT Roofnet provide access to their code, some of them without much documentation. ManetLab on the other hand provides both its source code and an easy-to-install binary file, with documentation and examples online.

Specialized hardware. While most simulators can be easily deployed on almost any computer, many testbeds require specialized hardware. This requirement makes it harder for other researchers to install the testbed and execute existing protocols. Moreover, some testbeds are deployed in specific lab settings and allow remote users to connect, such as CMUTrueMobile (based on the Emulab testbed [31]), which offers access to its testbed built on custom robots, or ORBIT which offers a testbed of 400 fixed WiFi devices placed in a grid formation on the ceiling of a single room. Other tools are devised to use special hardware, such as MiNT-m that uses Roomba vacuum cleaner robots as underlying hardware in order to support mobility and custom hardware on which to run protocols. Airplug-emu is another such example and is designed to emulate vehicular network and runs on laptops connected to specific GPS and radio receivers. Remote solutions have the advantage of side-stepping the tool deployment stage, and often allowing node mobility, but they impose restrictions on the execution scenarios. Other tools along with ManetLab can be deployed on standard equipment, which makes it easier for others to deploy and evaluate them. These tools include: Castadiva, MobileEmu, mLab and Airplug-emu.

8.6 Conclusion

Even though performance evaluation is central when it comes to designing robust MANET-specific communication protocols, we believe this problem has not been addressed in a satisfactory manner so far. Either protocols were evaluated through simulations and the results might not be valid in a real MANET environment, or they were evaluated in a customized testbed, which makes it hard to reproduce experiments. In this paper we presented ManetLab as a solution to this conundrum: ManetLab aims at offering the best of both worlds, i.e., accurate and reproducible results. In future work, we plan to extend ManetLab to iOS devices and to add an offline control mode.

References

[1] Mustafa Al-Bado, Cigdem Sengul, and Ruben Merz. What details are needed for wireless simulations? - a study of a site-specific indoor wireless model. In *INFOCOM'12*, pages 289–297, 2012.

[2] Leonard Barolli, Makoto Ikeda, Fatos Xhafa, and Arjan Durresi. A testbed for manets: Implementation, experiences and learned lessons. *IEEE Systems Journal*, 4(2):243–252, 2010.

[3] R. Barr, Z.J. Haas, and R. van Renesse. Jist: An efficient approach to simulation using virtual machines. In *Software Practice & Experience, vol. 35, no. 6*, pages 539 – 576, 2005.

[4] R. Barr, Z.J. Haas, and R. van Renesse. Scalable wireless ad hoc network simulation. In *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks, chapter 19*, 2005.

[5] A. Buisset, B. Ducourthial, F. El Ali, and S. Khalfallah. Vehicular networks emulation. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pages 1 –7, aug. 2010.

[6] Benjamin A. Chambers. The grid roofnet: A rooftop adhoc wireless network. In *M.S. Thesis, MIT, Cambridge, Massachusetts, June 2002.*

[7] Xinjie Chang. Network simulations with opnet. In *Winter Simulation Conference Proceedings*, pages 307–314, 1999.

[8] Qi Chen, Felix Schmidt-Eisenlohr, Daniel Jiang, Marc Torrent-Moreno, Luca Delgrossi, and Hannes Hartenstein. Overhaul of ieee 802.11 modeling and simulation in ns-2. In *Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, MSWiM '07, pages 159–168, New York, NY, USA, 2007. ACM.

[9] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: An Overlay Testbed for

Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):00–00, July 2003.

[10] Pradipta De, Ashish Raniwala, Srikant Sharma, and Tzi cker Chiueh. Mint: A miniaturized network testbed for mobile wireless research. In *In The 24 th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2731–2742, 2005.

[11] B. Garbinato, A. Holzer, and F. Vessaz. Six-shot multicast: A location-aware strategy for efficient message routing in manets. In *Network Computing and Applications (NCA), 2010 9th IEEE International Symposium on*, pages 1 –9, july 2010.

[12] Benoît Garbinato, Adrian Holzer, and François Vessaz. Six-shot broadcast: A context-aware algorithm for efficient message diffusion in manets. In *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I on On the Move to Meaningful Internet Systems:*, OTM '08, pages 625–638, Berlin, Heidelberg, 2008. Springer-Verlag.

[13] Benoît Garbinato, Adrian Holzer, and François Vessaz. Context-aware broadcasting approaches in mobile ad hoc networks. *Comput. Netw.*, 54(7):1210–1228, May 2010.

[14] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, New York, NY, USA, 1999. ACM.

[15] Thomas R. Henderson, Mathieu Lacage, and George F. Riley. Network simulations with the ns-3 simulator. In *In Proceedings of SIGCOMM'08*, 2008.

[16] Jorge Hortelano, Juan-Carlos Cano, Carlos T. Calafate, and Pietro Manzoni. Testing applications in manet environments through emulation. *EURASIP J. Wirel. Commun. Netw.*, 2009:47:1–47:9, May 2009.

[17] Nacher M. Cano J.-C. Calafate C. Manzoni P. Hortelano, J. Virmanel: a mobile multihop network virtualization tool. In *Proceedings of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications, 2007. PIMRC 2007.*, pages 1 – 5, 2007.

[18] H. Tewari J. Flynn and D. O'Mahony. Jemu: A real time emulation system for mobile ad hoc networks. In *Proceedings of the First Joint IEI/IEE Symposium on Telecommunications Systems Research*, 2001.

[19] Weirong Jiang and Chao Zhang. A portable real-time emulator for testing multi-radio manets. In *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, pages 169–169, Washington, DC, USA, 2006. IEEE Computer Society.

[20] David Johnson, Tim Stack, Russ Fish, Dan Flickinger, Rob Ricci, and Jay Lepreau. Truemobile: A mobile robotic wireless and sensor network testbed?, flux technical note ftn-2005-02. In *In The 25 th Annual Joint Conference of*

the IEEE Computer and Communications Societies. IEEE Computer Society, 2006.

[21] A. Karygiannis and E. Antonakakis. mlab: A mobile ad hoc network test bed. In *Proceedings of the 1st Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing in conjunction with the IEEE International Conference in Pervasive Services*, 2005.

[22] Glenn Judd-Eric Anderson Kevin Borries, Xiaohui Wang and Peter Steenkiste. Experience with a wireless network testbed based on signal propagation emulation. In *Proceedings of IEEE European Wireless Conference (EW'10)*, 2010.

[23] Markku Kojo, Andrei Gurtov, Jukka Manner, Pasi Sarolahti, Timo Alanko, and Kimmo Raatikainen. Seawind: a wireless network emulator. In *In Proceedings of 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems*, 2001.

[24] M. Kropff, T. Krop, M. Hollick, P.S. Mogre, and R. Steinmetz. A survey on real world and emulation testbeds for mobile ad hoc networks. In *In Proceedings of the 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM'06)*, 2006.

[25] Elis Kulla, Makoto Ikeda, Leonard Barolli, Fatos Xhafa, and Jiro Iwashige. A survey on manet testbeds and mobility models. In James J. (Jong Hyuk) Park, Han-Chieh Chao, Mohammad S. Obaidat, and Jongsung Kim, editors, *Computer Science and Convergence*, volume 114 of *Lecture Notes in Electrical Engineering*, pages 651–657. Springer Netherlands, 2012.

[26] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, October 2005.

[27] M. Matthes, H. Biehl, M. Lauer, and O. Drobnik. Massive: An emulation environment for mobile ad-hoc networks. In *Wireless On-demand Network Systems and Services, 2005. WONS 2005. Second Annual Conference on*, pages 54 – 59, jan. 2005.

[28] Othman O. Khalifa Liana K. Qabajeh Mohammad M. Qabajeh, Aisha-Hassan A. Hashim and Jamal I. Daoud. Performance evaluation in manets environment. *Australian Journal of Basic and Applied Sciences, 6(1):143-148*, 2012.

[29] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping 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, New York, NY, USA, 1999. ACM.

[30] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 3, pages 1664 – 1669 Vol. 3, march 2005.

[31] T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *In Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM'06)*, pages 1 – 12, 2006.

[32] Andràs Varga. The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference (ESM'2001)*, June 2001.

[33] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *in Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.

[34] Yongguang Zhang and Wei Li. An integrated environment for testing mobile ad-hoc networks. In *Proceedings of the third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc?02)*, 2002.

[35] Junlan Zhou, Zhengrong Ji, Maneesh Varshney, Zhiguo Xu, Yi Yang, Mahesh Marina, and Rajive Bagrodia. Whynet: a hybrid testbed for large-scale, heterogeneous and adaptive wireless networks. In *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*, WiNTECH '06, pages 111–112, New York, NY, USA, 2006. ACM.

Chapter 9
Conclusion

This thesis being written as a collection of papers, each chapter already comes with its own conclusion, summarizing the key points of its content. Thus, the purpose of this conclusion is twofold: first to highlight and sum up the contributions of this thesis (Section 9.1), and secondly to give an idea of the current research perspectives opened by the various contributions of this thesis (Section 9.2).

9.1 Contributions Synthesis

In this thesis, we provide both conceptual and practical contributions to the field of distributed systems in general and more specifically to the field of MANETs. Our conceptual contributions consist of various algorithms used to route messages inside a MANET. Our practical contribution consist in a tool to evaluate the performances of such algorithms.

Dissemination algorithms. We surveyed existing algorithms and provide new ones for broadcasting and multicasting messages in MANETs. After describing both our algorithms and several existing ones in the same format, in order to compare their structures and mechanisms, we proposed a classification of those broadcast algorithms. Our classification relies on four categories: *context-oblivious* characterizes deterministic or probabilistic algorithms with no context information, *network traffic-aware* characterizes algorithms monitoring the network to determine if the number of messages retransmissions is sufficient to ensure reliability, *power-aware* characterizes algorithms using signal power to decide if messages have to be retransmitted or not, and *location-aware* characterizes algorithms using location to decide if messages have to be retransmitted or not. Using simulations, we compared their respective performances. Our results show that context-oblivious algorithms are very simple but not very efficient, yet they provide the advantage of a quick message propagation in the network. Network traffic-aware algorithms are also very simple but they exhibit a higher degree of efficiency. Similarly, location-aware algorithms further increase efficiency, but require the ability of the device to determine its location. Power-aware approaches are less efficient, but exhibit the best reliability among the *best-effort* algorithms, especially in low densities of nodes. So, taking advantage of context-aware information, such as location or signal power, can help improve the performance of dissemination algorithms.

Adaptive algorithms. The behaviour of those dissemination algorithms – as well as their performances – differs significantly depending on the nodes density. An *adaptive algorithm* can dynamically change its own settings at runtime, such as the probability to retransmit a message or the threshold of message retransmissions used to discard the propagation of a message. An adaptive algorithm allows a node to choose the best settings for the nodes density sensed by this node. Thus, an adaptive dissemination algorithm can further improve performance compared to a non-adaptive one.

Saving energy. As MANETs are often composed of mobile devices with constrained resources, so power consumption is a key issue of those dissemination algorithms. To modulate the transmission range of nodes, we use a *power-law* approach. Nodes set their signal strength using a power-law distribution, which result in a few nodes having a large transmission range and all the other nodes having a small transmission range. We show that by modulating the transmission range of the nodes, we can save energy with virtually no negative impact on effectiveness.

Performance evaluation tools. To evaluate the performance of dissemination algorithms in MANETs, we can either use simulators or testbeds. One of the main difference between simulations and testbeds is the absence of physical layers, i.e., radio signal, in simulations. So simulators have to model radio signal, its propagation, interference, physical obstacles, etc. This may lead to decreased accuracy of simulators results. On the other hand, simulators are widely used by the research community and their source code is generally accessible online, which makes simulation-based evaluations fairly reproducible. Testbeds, on the contrary, rely on real mobile ad hoc networking and therefore tend to offer a high-level of accuracy. For this very reason however, they also tend to impose a high development and deployment cost. To address these issues, we introduced ManetLab, a testbed framework. ManetLab offers the inherent accuracy of testbeds and also offers high reproducibility of simulators thank to its plugin architecture and to its standardised hardware. We compared performance evaluation results of broadcast algorithms conducted with ManetLab to those of two simulators. We found that in simple environments (few interference), simulators results match testbeds results. However simulators results do not match testbeds results in more complex environments. So, we believe that ultimately accuracy and reproducibility can only be achieved by combining simulations and testbeds for performance evaluations of dissemination algorithms.

9.2 Perspectives

Moving beyond simulations. Context-aware dissemination algorithms have been broadly studied by the research community. Chapter 2 is our own survey, and performance comparison based on simulations, of such algorithms. But there exist many more routing algorithms in MANETs.[1] Many of those algorithms and protocols are rarely implemented. Most of them (up to 75% according to [2]) use simulations for performance evaluation. Although simulations are interesting and necessary, we must not forget that the ultimate goal is the deployment and use of such technology in the "real world". By relying exclusively on simulations to evaluate the performance of MANET-specific algorithms, there is a risk that the produced algorithms move away from reality although they perfectly match the models in the simulation "sandbox". In addition, for several reasons given among others by [2] and by Chapter 8, simulations have not only their usefulness but also their limitations. So we believe that the MANET research community should seriously consider moving beyond simulations.

Improving ManetLab. To move towards what we propose in the previous paragraph, we implemented a testbed framework for MANETs. Implementing an ad hoc algorithms or protocols in a real MANET is a difficult task because it requires to access the lowest levels of the system, e.g., MAC layer, and because testing on several geographically distributed devices is a time consuming task without adequate tools. ManetLab is our contribution to help developers of such algorithms. ManetLab is an open-source testbed framework for MANETs, which provides (1) a library to facilitate access to the MAC layer, (2) a deployment tool to distribute the protocol code to several remote devices, and (3) a plugin architecture to share this code with the community. ManetLab is currently in its first version. One track of future work will consist in improving ManetLab by adding the following features.

- We need to offer an *offline mode* that allows developers to use a real MANET and collect feedback without requiring a second network interface to transmit feedback to the ManetLab controller. This will make it possible to use ManetLab on mobile devices such as smartphones and tablets.
- We need to improve analysis tools to help ManetLab users understand and give meaning to collected data. ManetLab misses features like GUI analyzing tools or the ability to "replay" a previous execution in a graphical tool, in the same way NetAnim[2] does for NS simulators or the Instruments application for OS X.

[1] `http://scholar.google.com` offers thousands of references. We ourselves contributed to propose or optimize such algorithms (Part II of this thesis).

[2] `http://www.nsnam.org/wiki/index.php/NetAnim`

ManetLab would also benefit from a better integration with packet analyzer tools like Wireshark.[3]

- Finally, a web portal would be useful in order to promote ManetLab and to create a community sharing their MANET-specific algorithms.

Defining new scenarios. As described in the introduction of this thesis (Chapter 1), only a few MANET implementations are in production today, due both to the fact that it is technically hard to manage a decentralized network and to the fact that operators have found no satisfactory economic model yet. Finding use cases where it is appropriate to use MANETs, and implementing MANETs in those cases, will help to increase knowledge and practice of MANETs. Hopefully MANETs will eventually become easier to deploy and manage. Researchers have often described scenarios involving MANETs as infrastructureless communication for disaster or war situations. These are not the only scenarios in which MANETs may help to improve communications. Hereafter we give two examples of MANET usage that seem more promising than ever.

- **Self-configuring and peer-to-peer networks.** The development of the IEEE 802.11 standards, e.g. IEEE 802.11n, has brought wireless cards with multiple antennas to the market. One can thus envision devices connected to both an infrastructure-based network and a MANET. Infrastructure-based networks are used for the *traditional* connection to the Internet we know today, while the MANET may be used to exchange files, synchronise data or discover nearby devices, and to configure them without unnecessarily loading the infrastructure-based network. This use of MANET is also promising for mobile payment or ticketing solutions. Indeed, the customer's mobile device can form a MANET with the seller's terminal to communicate in order to make the financial transaction.
- **Swarm robotic.** An emerging trend in robotics consists in using swarms of simple robots to achieve a complex task. This trend takes its inspiration from the observation of colonies of insects where a complex collective behaviour emerges from simple individual behaviours. MANETs are an ideal way to communicate inside the swarm, to allow robots to coordinate. MANETs have the advantage to be fully decentralized and infrastructureless. So if a robot fails, it will not impact the whole system.

Towards the *Internet of things*. The two previous scenarios have in common that they connect several nearby devices together. This is the vision of the *Internet of things* [1] where all objects, simple or complex, can be connected together. So users are not anymore required to have a specific device to be connected (the "old" computer), but any object around them can serve as a gateway to the Internet.

[3] `http://www.wireshark.org`

This is also known as *ubiquitous* or *pervasive* computing. Examples of such objects are already present or will soon be available. The following examples exceed the traditional example of the *Internet refrigerator*.[4]

- **Hardware.** Cheap and small open-source hardware allows to create connected objects. Such hardware can be connected to sensors and has IEEE 802.11 capabilities. For example, the Raspberry Pi[5] has an educational purpose while the Flyport[6] targets companies.
- **Games.** At CES 2013, Lego presented an upgrade of its Mindstorms kit – a programmable brick to create toy robots.[7] This brick can be connected both to sensors and to the Internet in a wired or wireless way.
- **Clothes and accessories.** Today, Google often demonstrates its Google Glass project, where glasses provide augmented reality, voice control and Internet connection.[8] Those glasses are already available in private beta for developers. At SXSW 2013, Google also presented a concept of *talking shoes*.[9]

To connect all those objects, MANETs – at least for nearby communications – are a good option as they offer a fault-tolerant and scalable communication solution. We can envision, in a world where nearly all objects are connected, objects uniquely identified by IPv6 communicating locally over a MANET. So metaphorically, MANETs may eventually become the "medium" used between objects in the era of the *Internet of things*.

References

[1] Kevin Ashton. That 'internet of things' thing. *RFiD Journal*, 22:97–114, 2009.
[2] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, October 2005.

◊◊◊

[4] In 2000, LG launched the first Internet refrigerator for $17'000. This high price was one of the reasons of its commercial failure.
[5] `http://www.raspberrypi.org`
[6] `http://www.openpicus.com`
[7] `http://mindstorms.lego.com`
[8] `http://www.google.com/glass/start`
[9] `http://www.theverge.com/2013/3/9/4083928/google-made-a-talking-shoe-for-sxsw-2013-video`