Thèse de doctorat

UNIVERSITÉ PIERRE ET MARIE CURIE
UPMC SORBONNE UNIVERSITÉS

École doctorale

EDITE DE PARIS
INFORMATIQUE, TÉLÉCOMMUNICATIONS ET ÉLECTRONIQUE

présentée par

Matteo Sammarco

pour obtenir le grade de

Docteur de l'Université Pierre et Marie Curie

Dissémination multi-contenus opportuniste : Monitorage passif et adaptation aux conditions du réseau

soutenue le 28/05/2014 devant le jury composé de :

Andrzej Duda	Rapporteur	Professeur Grenoble INP-Ensimag
Thierry Delot	Rapporteur	Professeur Université de Valenciennes
Olivier Marcé	Examinateur	Ingénieur de Recherche Alcatel-Lucent Bell Labs
Luigi Iannone	Examinateur	Maître de Conférence Télécom ParisTech
Giovanni Pau	Examinateur	Professeur UPMC Sorbonne Universités
Marcelo Dias de Amorim	Directeur	Directeur de Recherche CNRS

Numéro bibliothèque :_____

PhD Thesis

UNIVERSITY PIERRE AND MARIE CURIE
UPMC SORBONNE UNIVERSITÉS

Doctoral school

EDITE DE PARIS
COMPUTER SCIENCE, TELECOMMUNICATIONS, AND ELECTRONICS

presented by

Matteo Sammarco

submitted in partial fulfillment of the requirements for the degree of

Doctor of Science of the University Pierre and Marie Curie

Opportunistic multi-content dissemination: Passive monitoring and adaptation to network conditions

Commitee in charge:

Andrzej Duda	Reviewer	Professor Grenoble INP-Ensimag
Thierry Delot	Reviewer	Professor Université de Valenciennes
Olivier Marcé	Examiner	Research Engineer Alcatel-Lucent Bell Labs
Luigi Iannone	Examiner	Assistant Professor Télécom ParisTech
Giovanni Pau	Examiner	Professor UPMC Sorbonne Universités
Marcelo Dias de Amorim	Advisor	CNRS Research Director

Acknowledgements

I want to express my complete gratitude to Marcelo Dias de Amorim, *deus ex machina* of my thesis. I hope to have learnt several things from him, in the research area and in everyday life. Nevertheless, I still think that Maradona has been a stronger player than Zico.

I wish to thank also the reviewers of my first manuscript version and the other members of the jury, since they posed me many constructive remarks.

I spent in LIP6 wonderful years. I must thank for this my colleagues: Nadjet (Nad: jet, jet), Tiphaine (the tea master girl), Adisorn (the asian kind koala), Raul (he's the guy who made me laugh for a whole day, continuously), Mubashir, Alex, Filippo, Yesid, and more recently Salah (puré!), Ahlem, Fadwa, and The Three Musketeers Alex (moquette hair), Benjamin (the pourquoi guy), Quentin (the one who eats many of cakes).

In many interesting moments spent in the lab, Miguel was there. I really appreciate his ideas and all the discussions we had. I wish him the best, as long as to Renata and to Gabriel. Remaining in the South America area, I want also to thank Luis, for his adding value remarks.

The LIP6 could not be the same without Prom. We all feel his presence when he's around, and we miss him when he's not. I spent amazing days in Hong Kong with him. Equally for Marguerite.

I must also thank the friends I met during these years in France: Silvio (I still remember that ragù he cooked at Easter), Paolo (he introduced me Emacs), Gege (he introduced me to the Mix), Giuseppe (I remember thoughts we shared in Agnano before going to France), Marianna (the friend who everyone wish to have), Max (Gallo, but not the novelist), Jessica (famous for her board games collection), Marzia (she's always so dynamic), Mattia, Stefano, Davide, Sharon, and Claudione.

Finally I must thank all my family and, in particular, Pina who fills all my days with love and happiness.

I'm really grateful to all these people. If I forgot some, I will offer them a pizza one day.

Résumé

La pénétration du marché des appareils mobiles, leurs fonctions, ainsi que la multiplicité des applications disponibles ont connu une croissance impressionnante ces dernières années. Par conséquent, les smartphones, les tablettes et les ordinateurs portables sont devenus à la fois producteurs et consommateurs de contenus générés par les utilisateurs. Ces appareils mobiles motivent également de nouveaux paradigmes de communication tels que la possibilité d'établir, de manière opportuniste, des liens directs de dispositif à dispositif lorsque deux nœuds mobiles entrent dans la portée sans fil de l'autre.

Les communications opportunistes permettent une couverture étendue dans les endroits où il n'existe aucune infrastructure réseau disponible et des stratégies de délestage de données pour aider les opérateurs à soulager la charge de leurs infrastructures.

Dans cette thèse, nous considérons le cas de la diffusion opportuniste de plusieurs grands contenus d'un point de vue expérimental. Cela implique de revisiter, entre autres, l'hypothèse selon laquelle les contacts entre nœuds mobiles ont une capacité suffisante pour transférer n'importe quelle quantité de données.

Dans la première partie de cette thèse, nous commençons par implémenter EPICS dans des terminaux Android. EPICS est un protocole réseau spécialement conçu pour l'échange opportuniste de grands contenus. Nous procédons à une évaluation de ses performances dans de nombreux scénarios. Bien que les résultats aient révélé de bonnes performances pour quelques contenus, le système montre des limitations lors du passage à l'échelle. Malheureusement, les logs de la couche applicative, enregistrés lors de l'expériences, ne sont pas suffisants pour comprendre ces résultats inattendus. Nous proposons alors de s'appuyer sur la surveillance passive du trafic et sur l'analyse des traces sans fil pour déterminer les limites et les possibilités d'amélioration. Cette méthodologie nous suggère de mieux exploiter la dynamique de la topologie du réseau par DAD, un nouveau protocole réseau pour la diffusion de contenus, qui envoie une rafale de paquets de données de façon adaptative au lieu de la stratégie d'EPICS de transmission par fragments. Nous comparons les deux protocoles expérimentalement et, à l'aide des traces de contacts, soit réelles, soit synthétiques, nous obtenons des gains importants avec cette nouvelle approche.

La surveillance passive est une partie essentielle de notre travail et nous avons décidé d'approfondir la question du passage à l'échelle d'un tel système. La deuxième partie de cette thèse traite donc de la façon d'aborder le problème du passage à l'échelle des systèmes

de surveillance de réseau local sans fil existants. Cela nous permet de procéder à une mesure expérimentale plus étendue. Nous proposons deux approches originales. Avec la première, basée sur la similarité des traces et des algorithmes de détection de communautés, nous sommes en mesure de déterminer le nombre de moniteurs nécessaires dans une zone géographique cible et leur placement. D'autre part, compte tenu d'une flotte de moniteurs, le même procédé peut être utilisé pour étendre la zone sous surveillance. La deuxième approche est basée sur des mesures collaboratives. Dans ce cas, nous considérons le risque de mesures biaisées en raison d'attaques d'utilisateurs malveillants qui peuvent générer des traces fallacieuses. Nous proposons ensuite une méthode pour détecter ces comportements malveillants en utilisant l'analyse de graphes basée sur les traces recueillies.

Mots-clefs

Réseaux opportunistes, réseaux mobiles à connectivité intermittente, caractérisation des contacts, dynamique des réseaux, analyse basée sur les graphes, dissémination de contenus multimédia, pair-à-pair, monitorage du trafic IEEE 802.11.

Abstract

The market penetration of mobile devices, their hardware capacities, as well as the multiplicity of available applications have experienced an impressive growth in the latest years. As a consequence, smartphones, tablets, and laptops have become both producers and consumers of user-generated contents. They also motivate novel communication paradigms such as the possibility to establish, in an opportunistic fashion, direct device-to-device links whenever two mobile nodes enter within the wireless range of each other.

Value-adds of opportunistic communications range from extended coverage where there is no network infrastructure available to the realization of offloading strategies to help operators relieve the load in their infrastructures. In this thesis, we consider the case of opportunistic dissemination of multiple large contents from an experimental point of view. This implies revisiting, among others, the common assumption that contacts have enough capacity to transfer any amount of data.

In the first part of this thesis, we start from an Android implementation of EPICS, a network protocol especially designed for exchanging large contents in opportunistic networks, on off-the-shelf devices. We conduct an experimental campaign evaluating its performance in many scenarios. Although the results revealed good performance for a few contents, the system shows severe limitations when scalability comes at play. Unfortunately, application-level logs stored during the experimentation are not enough to understand unexpected results. We propose then to rely on passive traffic monitoring and wireless traces analysis to find out limitations and uncovered improving possibilities. This methodology suggests us to better exploit the dynamics of the network topology through DAD, a new content dissemination protocol that adaptively sends bursts of data instead of the per-fragment transmission strategy of EPICS. We compare both protocols experimentally and using synthetic contact traces and show significant gains of the proposed approach.

Passive monitoring is an essential part of our work and we decided to investigate further some issues that remained open when we performed our experiments. The second part of this thesis deals with how to tackle the scalability problem of legacy WLAN monitoring systems in order to conduct a wide area experimental measurement. We propose two original approaches. With the first one, based on trace similarity and community detection algorithms, we are able to identify how many monitor we need in a target area and where to place them. On the other hand, given a fleet of monitors, the same method can be used

to stretch the area under observation. The second approach in based on collaborative measurements. In this case we face the risk of biased measures due attacks of malicious users generating adulterated traces. We then propose a method to detect such malicious behaviors by using graph-based analysis of collected traces.

Keywords

Opportunistic networks, intermittently-connected mobile networks, contact characterization, network dynamics, graph-based analysis, multimedia content dissemination, peer-to-peer, IEEE 802.11 traffic monitoring.

Contents

Résumé III

Abstract V

Contents VII

1. Introduction 1
1.1. Context and problem definition . . . 1
1.2. Storyline and contributions . . . 4
1.3. Workflow . . . 5
1.4. Outline of the manuscript . . . 6

I Refining opportunistic content dissemination: Strategies, measurements, and experimental evaluation 9

2. Background on opportunistic multi-content dissemination 11
2.1. Background . . . 11
2.2. Problem statement . . . 13
2.3. PACS: Intra-content selection strategy . . . 15
2.4. EPICS: Inter-content selection strategy . . . 16
2.5. Summary . . . 18

3. PePiT: An Android-based substrate for multi-content dissemination 19
3.1. Requirements . . . 19
3.2. Architecture . . . 20
3.3. Internal data structures . . . 22
3.4. PePiT settings menu . . . 23
3.5. Deployment on Android mobile devices . . . 23
3.6. An illustrative scenario . . . 24
3.7. Experimental setup . . . 25
3.7.1. Experimental parameters . . . 25
3.7.2. Discussion . . . 26
3.7.3. Benchmarking . . . 26
3.8. Experimental results . . . 26
3.8.1. EPICS in an emulated mobile scenario . . . 29
3.8.2. EPICS in a mobile scenario . . . 30
3.9. Summary . . . 31

4. DAD: Bringing dynamics to EPICS **33**
4.1. Rationale . . . 33
4.2. EPICS breakdown . . . 34
4.2.1. Impact of the piece size . . . 35
4.2.2. Impact of the transport layer protocol . . . 36
4.2.3. Impact of the burst size . . . 37
4.3. DAD: Dynamically Adaptive Dissemination . . . 39
4.3.1. Room for improvement . . . 39
4.3.2. DAD: Bringing dynamics to EPICS . . . 43
4.4. Summary . . . 44

II WLAN monitoring: Basics, deployment, and collaborative behavior **45**

5. IEEE 802.11 traffic monitoring: Background and problem statement **47**
5.1. Legacy monitoring methods . . . 47
5.1.1. Large deployments . . . 48
5.1.2. Trace inference . . . 49
5.2. IEEE 802.11 background . . . 49
5.3. Detailed problem statement . . . 52
5.3.1. Merging module computational complexity . . . 53
5.3.2. Biased measures due to corrupted traces . . . 54
5.4. Summary . . . 54

6. Scalable wireless traffic capture **57**
6.1. Improving Trace Selection . . . 59
6.2. Experimental Setup . . . 61
6.2.1. Scenarios . . . 61
6.2.2. Merging tool . . . 63
6.2.3. Trace Similarity . . . 64
6.3. Community detection . . . 68
6.3.1. Algorithms . . . 70
6.3.2. Results for community detection . . . 70
6.4. Trace ranking . . . 71
6.5. Evaluation . . . 73
6.5.1. Proposed strategy vs. trace size . . . 73
6.5.2. Proposed strategy vs. node degree . . . 74
6.6. Summary . . . 76

7. Sensitivity to input traces **79**
7.1. Detecting vulnerabilities . . . 80
7.1.1. Accuracy . . . 83
7.1.2. Detection system . . . 84
7.2. Experimental Setup . . . 85
7.3. Accuracy Measurements . . . 86
7.3.1. Individual and merged traces . . . 86
7.3.2. Number of traces . . . 87
7.3.3. Position of sensing nodes . . . 88
7.4. Impact of Attacks . . . 90
7.5. Detecting Potential Attackers . . . 91
7.6. Summary . . . 95

8. Conclusion and perspectives **97**
8.1. Conclusion 97
8.2. Perspectives 98

Appendices **101**

A. Résumé de la thèse en français **103**
A.1. Introduction 103
A.2. Définition du problème 104
A.3. Contribution 1 : PePiT, un substrat basé sur Android pour la diffusion multi-contenu 106
A.3.1. Évaluation 106
A.4. Contribution 2 : DAD, EPICS dynamique 108
A.4.1. Évaluation 109
A.5. Contribution 3 : Passage à l'échelle de systèmes de surveillance passive . . . 110
A.5.1. Évaluation 110
A.6. Contribution 4 : Sensibilité aux traces d'entrée 112
A.6.1. Système de détection 112
A.6.2. Évaluation 113
A.7. Conclusions et perspectives 113
A.8. Perspectives 115

B. PePiT: code snippet and UML class diagram **119**

C. List of publications **123**
C.1. Published 123
C.2. Under review 123
C.3. Other publications 124

Bibliography **125**

List of Figures **135**

List of Tables **139**

Chapter 1

Introduction

The growth of traffic from wireless and mobile devices has more than optimistic forecasts[4]. In this direction, together with innovative mobile development frameworks, new opportunistic content sharing applications are catching on[7;16;70;72;75;110]. Such applications fulfill to the current societal demand to produce and consume larger and larger user-generated contents (UGCs)[23] according to the triple-A paradigm (Anywhere, Anytime, Any device).

The compatibility of mobile wireless devices to the triple-A paradigm justifies the effort lavished by the research community about opportunistic networks in the latest years. Opportunistic networks are created by sporadic and direct contacts among mobile users. This peculiarity makes them suitable to exchange contents in many contexts and environments such as: local and temporary events, disaster recovery and crowded places, vehicular and sensor networks, satellite and pocket-switched networks[20;39;43;44;51;76;121]. This kind of network is also sometimes identified as Delay/Disruption Tolerant Networks (DTNs) or Intermittently Connected Mobile Networks (ICMNs).

Understanding the dynamics at the application level and underlying wireless standard mechanisms becomes fundamental to design an efficient content exchange mechanism, especially in crowded environments or in situations where short contact windows are the rule.

In this thesis, we address such problems by linking the performance of opportunistic content sharing applications with the surrounding wireless traffic. Not diminishing the importance of simulations, only in conditions very close to reality we can face representative issues. For this reason, we have deliberately adopted an experimental approach to base our analysis on real applications deployed on off-the-shelf devices.

1.1. Context and problem definition

Users usually retrieve contents from the Internet or ask for services through the traditional client-server model. Since the Internet has been designed following the end-to-end principle, a content is identified by a Unified Resource Locator (URL) associated with a host

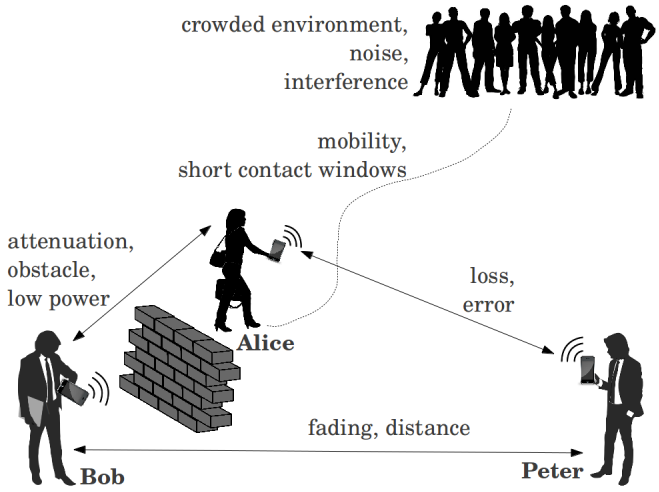


Figure 1.1: Problems occurring in wireless communications.

or web servers. This model works smoothly until some contents become so popular that links connecting to the server and its computational power become a bottleneck.

Solutions such as peer-to-peer and content-distribution networks are efficient responses to this problem, but they depend on the availability of network infrastructure. In absence of the latter, other approaches propose to rely on direct message exchanges between nodes through wireless ad hoc networking. Ad hoc networks are, by definition, decentralized networks where nodes participate in routing and data forwarding in according to the dynamically changing network connectivity. Since no central unit is present, accessing the medium is not anymore regulated by a controller. If we further consider the intermittent and opportunistic connectivity with the possibility of long transmission delays, sparse and heterogeneous nodes, we deal then with Delay-Tolerant Networks (DTNs)[22].

To illustrate such a scenario, let us suppose that Bob wants to transmit a message to Peter using opportunistic communications as shown in Figure 1.1. They are too far from each other to directly communicate. Bob could then send the message to Alice who is passing by. Various wireless communication problems may occur: attenuation by an obstacle, multipath, transmission errors, and message losses, to cite a few. If Alice manages to get the message, she stores and carries it until she jumps into the Peter's transmission range. Such a store-carry-and-forward mechanism is the general approach to route messages in DTNs.

Broadly speaking, routing protocols in DTNs must negotiate the tradeoff between protocol overhead and delivery performance. On one hand, flooding all messages tends to minimize the delivery delay at the cost of increased storage and transmission overhead. On the other hand, sending the message directly to the end recipient tears down the overhead along with the probability to receive the message within a short delay. This trade off is studied in many routing schemes that take into account mobility and node contact probability[11;19;45;63;65;104;109;119;120;122].

The DTN architecture is inherently node-centric with unicast message delivery that is independent from the underlying transport protocols[22]. Messages may be fragmented and fragments may be bundled together anywhere in the network. Chopping large contents into smaller pieces for a more effective dissemination in opportunistic sharing applications is in fact a natural approach to adopt in DTNs. Nevertheless, several problems (surprisingly under-considered in the literature) arise:

- Which content to transmit when a contact happens?
- Once the content selected, which piece (fragment) should be prioritized?
- How big should be a piece?
- Is it worth using a reliable transport protocol?
- Is it worth transmitting bursts of pieces before recomputing a new prioritization ranking?

We investigate all these aspects in the first part of this work. Other aspects such as localization privacy and security are out of the scope of this manuscript.

Opportunistic networks in practice. Despite the number of content distribution strategies elaborated for opportunistic networks [49;69], only few of them have been implemented on mobile devices. Haggle is a push-based framework that decouples the specific application business logic from the communication technology providing mechanism for late-binding interfaces[106]. In this way, upper-layer applications can agnostically use different communication modes (infrastructure, infrastructure-less, Bluetooth, ad hoc Wi-Fi). MobiClique[77], Opportunistic-Twitter[90], and Haggle-ETT[68] are examples of applications built on top of the Haggle framework. They provide, respectively, a mobile social networking middleware, an ad hoc twitting application, and an electronic triage tag system.

Similar to Haggle in purpose, but different in design, WiFi-Opp exploits the mobile Wi-Fi AP feature (tethering) to create opportunistic networking[110]. PodNet allows sharing podcasts and any files during opportunistic contacts[2]. Users must preliminarily subscribe to receive contents, which are organized into feed channels. 7DS introduces a new platform to develop mobile applications for disruption-tolerant mobile networks[71]. It provides a modular platform with transport- and application-layer functionalities for mobile nodes to exchange information in store-carry-and-forward mode. Proximiter allows sharing various types of content on portable devices through direct and multi-hop communications[115].

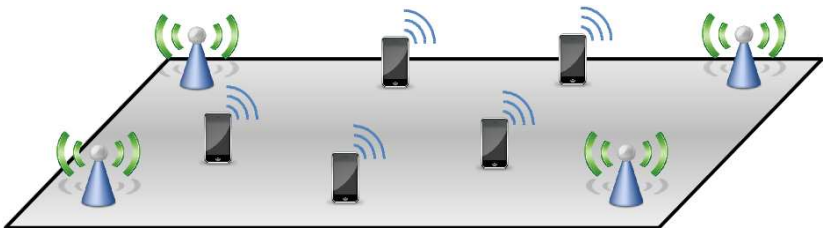


Figure 1.2: Wireless traffic capturing during opportunistic content exchange

1.2. Storyline and contributions

This thesis work started with the goal of filling the lack of implementation of *opportunistic content dissemination protocols*. In the context of the ANR Crowd project[1], we proposed, designed, and implemented an Android application called PePiT (Chapter 3). PePiT implements EPICS, an opportunistic multi-content dissemination protocol also proposed in the context of the ANR Crowd project.[2]

In a few words, EPICS executes two main actions. First, it chops contents into smaller chunks (or pieces) to achieve a more effective dissemination. Second, as an inter-content selection strategy, it selects, at each contact, which content and which piece of that content to exchange in order to have a fast dissemination. With this regard, our first contribution in the area of opportunistic content dissemination is then the following:

> **Contribution 1.** We develop PePiT, an Android opportunistic content dissemination application based on EPICS. Its modular architecture makes it suitable to be extended with minimum effort. It also successfully works on virtual Android-x86 systems[1] running on any host machine. Having a working application, we conduct an experimental campaign to evaluate the performance of EPICS.

Although our experiments with PePiT showed the fairness of EPICS, we noticed, out of the experimental results, a significant room for improvement. In fact, we observed that the limitations were due to wireless phenomena that we could not understand by examining only application-level logs. In order to have a more complete view and then find out all the obstacles for fast content dissemination, we decided to deploy a passive measurement system to capture the wireless traffic during the experiments, as shown in Figure 1.2.

This experimental campaign drove us to propose the following:

[1] http://anr-crowd.lip6.fr

[2] Although being part of a joint work, EPICS was reported in the Ph.D. thesis of Nadjet Belblidia, the main contributor in the design of the protocol[13]. Here, we focus on the implementation aspects, as well as the evaluation, of this proposal. For sake of completeness and clarity, however, we briefly describe EPICS in Chapter 2.

Contribution 2. Taking advantage of wireless traffic traces, we could then deeper analyze opportunistic content dissemination mechanisms *in vivo*. We exploited wireless captured traces as a support to analyze results. The results led us to propose **DAD**, standing for Dynamically Adaptive Dissemination which further improves the content diffusion latency. We have inspected the profit margin on real movement traces and on simulated traces generated by SIMPS simulator, a social-based mobility simulator that we developed, based on the SIMPS model[18].

At this point, we faced some limitations of the passive measurement system and decided to investigate this module further. In fact, we noticed that when the number of nodes involved in the experiment grows and they are mobile, the monitoring system quickly becomes insufficient with regard to spatial coverage and to accuracy of the captured traces. We propose solutions to solve these issues in two aspects. First, we consider the problem of monitoring coverage:

Contribution 3. We propose a new approach to monitor WLAN traffic based on trace similarity and community detection algorithms. The advantage of this approach is twofold. Given a target area, it gives the minimum number of monitors required to cover this area. The idea is extend the covered area without having to acquire extra monitors and increase maintenance/management overhead. In other words, given a monitor fleet, our proposal points out the monitor that has to move to extend the target area (preserving the capture quality).

Second, we tackle the problem of sensitivity of the measurement system to corrupted traces:

Contribution 4. We propose a second approach based on collaborative measures. In this case, users take part in the capture process in exchange of something (e.g., connectivity). If this method presents a clear advantage in terms of scalability, it is also open to biased measures generated by corrupted traces or malicious users. We formalize the problem as a security issue and present two possible attacks based on trace adulteration together with countermeasures to detect them.

1.3. Workflow

Finding the right configuration among several parameters in distributed and opportunistic content dissemination protocols is not an easy task. Both protocol-specific and

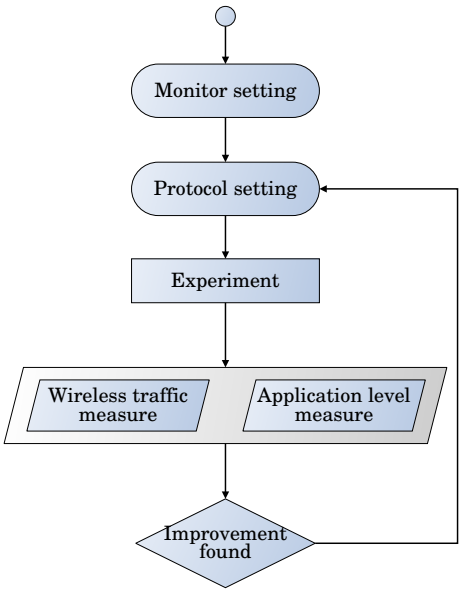


Figure 1.3: Experimental workflow.

communication-technology mechanisms must be taken into account to have a right view. We present in Figure 1.3 our complete experimental workflow. Once the experimental stage is chosen, we proceed with the monitor and the protocol settings. The monitor setting concerns finding the quantity of monitors needed to have sufficiently good capture and their position. The protocol setting involves tuning parameters for the specific protocol under test. In this way, we get both application- and data-link level measures. If the analysis of such measurements highlights possible improvements, protocol parameters are adjusted accordingly and we can start a new series of experiments. In Chapter 4 we use this kind of pipeline to conduct our experimental campaign.

1.4. Outline of the manuscript

This thesis is structured in two parts. For the sake of coherence, each part has its own context and background section. The first part is dedicated to the design, implementation, and evaluation of an opportunistic multi-content dissemination protocol. In Chapter 2, we provide the necessary background and describe our baseline dissemination protocol, namely EPICS[15]. In Chapter 3, we detail PePiT, an Android-based communication substrate over which we implement EPICS. We capitalize the experimental results finding out limitations of EPICS and designing a more efficient opportunistic content dissemination protocol that we call DAD (Dynamically Adaptive Dissemination). We present and evaluate the performance of DAD in Chapter 4.

To better understand some unexpected results during the experiments, we passively monitored the wireless traffic. While running the monitoring system, we observed a number of limitations that could prevent one to efficiently employ a fleet of passive monitors, where

to place them, and scale up the monitoring system. In the second part of this manuscript, we tackle these issues. Firstly, in Chapter 5, we recall some notions of the IEEE 802.11 standard that will be helpful thereinafter and then introduce the state-of-the-art techniques about capturing WLAN traffic. In Chapters 6 and 7, we present our solutions to solve the aforementioned problems. For the collaborative solution, we also present our approach to detect biased measures generated by malicious users.

Part I

Refining opportunistic content dissemination: Strategies, measurements, and experimental evaluation

Chapter 2

Background on opportunistic multi-content dissemination

Concert halls, stadium, bus waiting shelters, are just few examples of places where collocation creates common interests between people. In such locations, it could be useful to locally exchange photos, audio files, travel information or headline news, using what we always have with us: our smartphones. In the latest few years, these devices have witnessed a very quick evolution and a high wide-spreading market penetration. It is expected that traffic from wireless and mobile devices will surpass traffic from wired devices by 2016[4]. The reason of this success must be ascribed also to both the increasing computational power and the wide range of connectivity interfaces embedded, making them suitable to use in ad hoc and opportunistic networks. All the evident advantages of opportunistic networks (fault tolerance, locality, scalability, infrastructure offloading) can lead to a new widespread content-centric, web 2.0 or media-sharing mobile applications such as proximity chat, local social networks, video and photo sharing, folksonomy, or microblogs.

In this chapter, after the background and the problem statement sections, we present EPICS. EPICS it the opportunistic multi-content dissemination protocol that we implement and evaluate in the following of this work. It is based on the grey relational analysis for the inter-content selection strategy.

2.1. Background

In opportunistic networks, contacts may have insufficient capacity to transmit the required amount of data. Some pioneer works take into account bandwidth constraints[11;57]. They propose both message scheduling and dropping policies in order to optimize different performance metrics. Nevertheless, in these solutions, authors assume unicast transmissions using replication-based routing schemes. Their main objective is then to optimize the replication in order to achieve best per message performance.

Data broadcasting in opportunistic and ad hoc networks has been the subject of several works. Williams et al. classify approaches into four main categories: simple flooding, probability based, area based, and neighbor knowledge[114]. In addition, a new data dissemination category based on network coding emerged recently[9;33]. The main objective of all these solutions is to achieve an efficient dissemination while minimizing the number of transmissions in the network. This is done by selecting the best relay nodes among all the neighbors an infected node has. Nevertheless, all these approaches assume that any contact is long enough to transfer the data under consideration. Indeed, these solutions answer to the question of how to select relay nodes instead of answer the question of how to select which content and which piece(s) of that content to transfer once the relay node is selected.

Pitkänen et al. studied the impact of data fragmentation in one-to-one opportunistic network communications[78]. They considered two fragmentation strategies: reactive fragmentation and proactive fragmentation. In reactive fragmentation, the sender starts transmitting the data until it is interrupted by the link failure caused by the end of the contact. In proactive fragmentation, the source node divides the data into pieces of standard size (based on the expected average contact capacity). They concluded that the reactive fragmentation with predefined fragment boundaries allows significant improvements in one-to-one communications.

Fragmentation is also inspired by BitTorrent sharing mechanism[34;58;83;97]. Most of these adaptations, however, aim at constructing and maintaining an overlay network that enables multi-hop message routing. In other terms, nodes do not need to be direct neighbors to become peers.

Nadan et al. proposed SPAWN, a cooperative strategy for content downloading in vehicular networks[73]. The piece selection scheme is based on a proximity-driven strategy called rarest-closest. Such a strategy selects the rarest pieces and then ranks them based on the distance to the closest peer which has that piece. SPAWN constructs an application-layer overlay that does not limit the peer selection to the one-hop neighborhood. Hence, it needs an underlay routing protocol that maintains multi-hop routes between peers.

Other solutions implemented file swarming by only considering one-hop communications and uniformly-distributed random piece selection[36;62]. Nevertheless, they use network coding in order to mitigate the coupon collection problem by increasing piece heterogeneity. Finally, some papers presented different architectures to enable mobile peer-to-peer distribution of large contents[40;50]. In both architectures, contents are exchanged opportunistically when nodes are within communication range. However, the piece selection strategy differs. Jung and al. used the random selection strategy[50] whereas Helgason et al. presented an implementation of a sequential strategy using a pull-based architecture[40].

Data dissemination in opportunistic networks has been the subject of several studies in the latest years[46;53;124]. A promising application for data dissemination in opportunistic networks is enabling content sharing among users on the move. Several content sharing

systems have been proposed [34;54;73;83;97;103;118]. As previously mentioned, most of these solutions aim at constructing and maintaining an overlay network to support multi-hop message forwarding. In our work, we rather consider immediate communication capabilities of the users to disseminate data.

Other solutions implement file-sharing by only considering one-hop communications. Some of them rely on the publish/subscribe paradigm[52;64] while others are based on flat peer-to-peer systems[36;40;50;60–62]. Similarly to ours, all these solutions aim at making the content available to all users in the network. Nevertheless, the main difference is that these solutions are generally pull-based. In other words, users make the decision of the content to receive by proactively querying other peers about subscribed feed contents. In such a scenario, inter-content selection comes from the client request. Conversely, we consider a push-based system where contents to be disseminated are weighted to fulfill the defined dissemination policy objectives.

Problems addressed in caching, replication, and content placing schemes are also addressed in some works[11;17;48;57;87]. Indeed, contents to be stored at mobile nodes are generally dictated by a global policy. Because user resources are limited, a kind of content priority (or utility) is introduced to enable decision making among contents to be stored. Nevertheless, the objectives differ. In such schemes, content selection is done to save user memory resources whereas the aim of our work is to apply a global dissemination policy. Ioannidis et al. investigate the optimality and scalability of dynamic content distribution over mobile social networks[47]. In this work, mobile users subscribe to a dynamic-content distribution service and share any content updates they receive. The target scheme is push-based – when two users are in range, the user having the freshest version pushes it to the second one.

2.2. Problem statement

Whereas communication opportunities have limited duration and capacity due to short duration contacts and power saving technologies, users, conversely, generate, consume, and share contents that are becoming increasingly larger. In such a situation, opportunistic content-sharing solutions must be reformulated to support efficient dissemination of large contents. In particular, data must be sliced so that smaller pieces are transmitted separately; this leads to a better use of short-lived contacts and promotes progressive content dissemination. The first challenge is then to choose which piece of the content to send upon a contact. The problem becomes even more challenging when multiple contents flow in the network at the same time. In a nutshell, the goal is to design an efficient strategy for deciding at each contact:

1. *Which piece to transmit,*
2. *from which content.*

Table 2.1: Summary of variables in opportunistic content dissemination context.

Variable	*Definition*
$\mathbf{N}$	Set of nodes in the network
N	Number of nodes in $\mathbf{N}$
$\mathbf{C}$	Set of contents to be disseminated
C	Number of contents in $\mathbf{C}$
K_j	Number of pieces that compose content c_j
$\mathbf{a}_{n_i,j}$	Availability bitmap associated with content c_j at node n_i
$\mathbf{p}_{n_i,j}$	Prevalence vector associated with content c_j at node n_i

To address the problem described above, we have chosen EPICS, a distributed protocol to help nodes decide which is the best piece to transmit during a contact in order to achieve a predefined dissemination policy. EPICS chops contents into small pieces and shuffles them to speed up the dissemination. To decide which piece from which content should be sent, EPICS relies on the grey relational analysis with the goal to prioritize contents having some expected features (most popular, most recent, or most urgent content). EPICS follows a "prevalence" principle used in a companion protocol, namely PACS[14;15]. While PACS considered only the intra-content piece selection, EPICS addresses the general problem when multiple contents co-exist.

We describe how EPICS works in the following sections as long as the network model.

Assumptions. Let $\mathbf{N} = \{n_0, n_1, \ldots, n_{N-1}\}$ be the set of N nodes in the network. Nodes are mobile, but we do not assume any a priori knowledge of mobility patterns. We assume instead that all nodes in the network are interested in a set of contents $\mathbf{C} = \{c_0, c_2, \ldots, c_{C-1}\}$. Each content c_j is initially only available at a single data source. We do not make any assumption on the creation time of contents.

For each content c_j, the source chops the content into K_j pieces of equal size (the piece size can be determined to optimize communication opportunities[15]). Pieces are sequentially identified as $c_j = \{d_0, d_1, \ldots, d_{K_j-1}\}$. Nodes use their contact opportunities to get pieces, i.e., we assume that there is no infrastructure to help the dissemination process. Nodes can get pieces from the data source and from any other node in the network having it.

Each node n_i locally stores an *availability bitmap vector* $\mathbf{a}_{n_i,j} = \{a_0, \ldots, a_{K_j-1}\}$ and a *prevalence vector* $\mathbf{p}_{n_i,j} = \{p_0, \ldots, p_{K_j-1}\}$ both associated with every known content c_j. The availability bitmap vector $\mathbf{a}_{n_i,j}$ keeps track of c_j content pieces that the node n_i holds. It contains binary values associated to each piece, where $a_m = 1$ if the node n_i has piece d_m, and $a_m = 0$ otherwise. The goal of the prevalence vector is to give a local view of the prevalent pieces in the network. Initially, each node associates an empty prevalence vector to each content.

All the variables are summarized in Table 2.1.

Algorithm 1 n_i PACS strategy

while $contact_with(n_j)$ **do**
  $receive_from(n_j, \mathbf{a}_{n_j,0})$;
  $\mathbf{p}_{n_i,0} \leftarrow \mathbf{p}_{n_i,0} + \mathbf{a}_{n_j,0}$;
  if $(\mathbf{a}_{n_i,0} \wedge (\neg \mathbf{a}_{n_j,0}) \neq \emptyset)$ **and** $(initiate_connection_with(n_j))$ **then**
    $d_{s_{i\rightarrow j}} \leftarrow prevalence_selection_from((\mathbf{a}_{n_i,0} \wedge (\neg \mathbf{a}_{n_j,0})), \mathbf{p}_{n_i,0})$;
    $send_to(n_j, d_{s_{i\rightarrow j}})$;
  end if
  if $(\mathbf{a}_{n_j,0} \wedge (\neg \mathbf{a}_{n_i,0}) \neq \emptyset)$ **and** $(connection_initiated_by(n_j))$ **then**
    $receive_from(n_j, d_{s_{j\rightarrow i}})$;
    $\mathbf{i}_{d_{j\rightarrow i}} \leftarrow \{i_0, \ldots, i_{K_0-1}\}; i_k = 0, \forall k < K_0\ (k \neq s_{j\rightarrow i}), i_{s_{j\rightarrow i}} = 1$
    $\mathbf{a}_{n_i,0} \leftarrow \mathbf{a}_{n_i,0} \vee \mathbf{i}_{d_{j\rightarrow i}}$;
  end if
end while

2.3. PACS: Intra-content selection strategy

We briefly present PACS, an intra-content piece selection strategy relying on the prevalence principle[14;15]. The goal of PACS is to achieve fast content dissemination while keeping the overhead low. To this end, nodes passively keep track of the dissemination progress of each piece of contents, so that they can appropriately prioritize their transmissions without inducing additional communication overhead.

Since this part only addresses the intra-content selection problem, let us assume, for the sake of simplicity, that only a single content c_0 is available in the network. We consider the multi-content case in Section 2.4.

Initially, all nodes in $\mathbf{N}$, except the one where c_0 was produced, have neither prevalence nor availability vectors associated to content c_0 because they are not aware of the presence of c_0 in the network. Nodes create these vectors as soon as they receive the availability vector, $\mathbf{a}_{n_i,0}$, relative to the new content c_0 from a node n_i. When nodes n_i and n_j meet, they exchange their availability vectors $\mathbf{a}_{n_i,0}$ and $\mathbf{a}_{n_j,0}$. Node n_i (resp. n_j) computes $\mathbf{a}_{n_i,0} \wedge (\neg \mathbf{a}_{n_j,0})$ (resp. $\mathbf{a}_{n_j,0} \wedge (\neg \mathbf{a}_{n_i,0})$), which gives the candidate pieces to be transferred. They also update their prevalence vectors respectively as: $\mathbf{p}_{n_i,0} \leftarrow \mathbf{p}_{n_i,0} + \mathbf{a}_{n_j,0}$, and $\mathbf{p}_{n_j,0} \leftarrow \mathbf{p}_{n_j,0} + \mathbf{a}_{n_i,0}$. Among the candidate pieces to be transferred, nodes select the one with the lowest prevalence. In the case of a tie, a piece is chosen in a uniformly distributed random way. Let $d_{s_{i\rightarrow j}}$ be the piece sent by n_i to n_j and $d_{s_{j\rightarrow i}}$ be the piece sent by n_j to n_i. After one round of exchanges, nodes update their availability vectors as: $\mathbf{a}_{n_i,0} \leftarrow \mathbf{a}_{n_i,0} \vee \mathbf{i}_{d_{s_{j\rightarrow i}}}$, and $\mathbf{a}_{n_j,0} \leftarrow \mathbf{a}_{n_j,0} \vee \mathbf{i}_{d_{s_{i\rightarrow j}}}$, where $\mathbf{i}_{d_{s_{i\rightarrow j}}}$ and $\mathbf{i}_{d_{s_{j\rightarrow i}}}$ are K_0 element vectors with all positions set to 0 except the position relative to the piece just received, which is set to 1. Note that prevalence vectors have a limited influence at the beginning, but they gain importance as nodes move and exchange pieces. The steps achieved by node n_i are stated in Algorithm 1.

2.4. EPICS: Inter-content selection strategy

EPICS bases the selection of the availability vector to transmit at each encounter on the grey relational analysis (GRA), which is a method in grey system theory for analyzing discrete data series[31;32]. This method allows measuring the degree of approximation of given data series x_u according to the reference data series x_0.

In the following, we summarize the steps needed to apply relational analysis processing:

(a) Set up the reference data series x_0

$$x_0 = \{x_0(1), x_0(2), \ldots, x_0(M)\},$$

where M is the number of considered data (or metrics) in the analysis. $x_0(v)$ represents the most favored value of the vth data ($1 \leq v \leq M$).

(b) Define the comparison data series x_u

$$x_u = \{x_u(1), x_u(2), \ldots, x_u(M)\},$$

where $1 \leq u \leq S$ and S is the number of compared data series in the analysis.

(c) Compute the difference data series Δ_u

$$\Delta_u = \{\Delta_u(1), \Delta_u(2), \ldots, \Delta_u(M)\},$$

where $\Delta_u(v) = |x_0(v) - x_u(v)|$.

(d) Get the global maximum value $\Delta_{\max}$ and the global minimum value $\Delta_{\min}$ from all data series

$$\Delta_{\max} = \max_u(\max_v \ \Delta_u(v)),$$
$$\Delta_{\min} = \min_u(\min_v \ \Delta_u(v)).$$

(e) Obtain, for each data v in each data series u, the grey relational coefficient $\gamma_u(v)$

$$\gamma_u(v) = \frac{\Delta_{\min} + \varsigma\Delta_{\max}}{\Delta_u(v) + \varsigma\Delta_{\max}},$$

where ς is a coefficient value between 0 and 1. ς is used to compensate the effect of $\Delta_{\max}$. Generally, ς is set to 0.5.

(f) Compute the grey relational grade for each data series u

$$\Gamma_u = \sum_{v=1}^{M}(\gamma_u(v) \times w(v)),$$

where $w(v)$ is the weight of the vth data in each series (with, $\sum_{v=1}^{M} w(v) = 1$). If all data in series have the same weight, Γ_u becomes:

$$\Gamma_u = \frac{1}{M} \sum_{v=1}^{M} \gamma_u(v).$$

As mentioned above, the grey relational grade value Γ represents the degree of approximation to the reference data series x_0. A high Γ_u indicates that the values in the data series x_i are, in general, close to the most favored values.

(g) Sort the k values of Γ into descending order.

We set EPICS to ensure a fairer dissemination delay for all contents regardless their *creation times* and their *sizes*. To reach this objective, weights should take into account both freshness and content size. Therefore, we consider these two metrics in the grey relational analysis ($M = 2$). Because we set the piece size to a fixed value for all contents, content size is defined as the number of pieces. Freshness is defined as the creation time, i.e., the time at which the data source generates a given content.

In the following, we detail how EPICS operates. To get the same scale for both metrics, every node first normalizes the evaluated values. To this end, each node gets the current values of both metrics associated with all known contents, takes the maximum and minimum values, and rescales the values in the range $[0, 1]$. Next, every node defines the reference data series. Since contents that are fresher and/or larger take more time to be disseminated, they should get higher weights. Hence, x_0 is set to $\{1, 1\}$. Then, each node computes Γ values of contents. The same weight is assigned to both metrics ($w(1) = w(2) = \frac{1}{2}$) and set ς to 0.5. Based on Γ values, weights are assigned to each content. The weights are then used to define the probability of selection of the corresponding availability vector. Algorithm. 2 details the content selection strategy applied at each node. Note that, since x_u values are normalized, Δ_{max} is always equal to 1 and Δ_{min} is always equal to 0.

An illustrative example. At $t = t_0$, n_1 knows three contents $\{c_1, c_2, c_3\}$ and n_2 knows four contents $\{c_1, c_2, c_3, c_4\}$. Suppose that contents c_1, c_2, c_3, c_4 were created at times 10, 30, 50, and 60 and have a size of 3, 2, 4, and 1 piece(s), respectively. Before sending one of its availability vectors, n_2 performs the following steps. First, it sets up x_u data series associated with every content c_u. To this end, n_2 gets normalized values for both freshness and size metrics. As previously mentioned, freshness is defined as the time of content creation. Thus, freshness values are $\{10, 30, 50, 60\}$. After normalization, they become $\{0, 0.4, 0.8, 1\}$. Size is defined as the number of pieces. Hence, sizes are $\{3, 2, 4, 1\}$. After normalization, sizes equals $\{0.66, 0.33, 1, 0\}$. Hence, the data series are: $x_1 = \{0, 0.66\}$, $x_2 = \{0.4, 0.33\}$, $x_3 = \{0.8, 1\}$, and $x_4 = \{1, 0\}$, with data series x_u associated with content c_u. Second, n_2 computes

Algorithm 2 n_i EPICS

1: $x_0 = \{1, 1\}$
2: $\Delta_{max} = 1$;
3: $\Delta_{min} = 0$;
4: **while** $contact_with(n_j)$ **do**
5: **if not** sending() **and not** receiving() **then**
6: **for** u **in** $range(known_contents())$ **do**
7: $x_u \leftarrow get_normalized_freshness_and_size(c_u)$;
8: $\Delta_u \leftarrow compute_difference_data_series(x_0, x_u)$;
9: $\Gamma_u \leftarrow compute_grey_grade(\Delta_u, \Delta_{max}, \Delta_{min})$;
10: **end for**
11: $prob \leftarrow \{\frac{\Gamma_1}{\sum_u \Gamma_u}, \ldots, \frac{\Gamma_k}{\sum_u \Gamma_u}\}$,
$k = range(known_contents())$
12: $s \leftarrow weighted_random_selection_of_content(prob)$
13: $send_to(n_j, \mathbf{a}_{n_i,s})$;
14: **end if**
15: **end while**

the difference data series Δ_u according to the reference series $x_0 = \{1, 1\}$: $\Delta_1 = \{1, 0.33\}$, $\Delta_2 = \{0.6, 0.66\}$, $\Delta_3 = \{0.2, 0\}$, and $\Delta_4 = \{0, 1\}$. Third, n_2 sets $\Delta_{max} = 1$ and $\Delta_{min} = 0$ and obtains the grey relational coefficients: $\gamma_1 = \{0.33, 0.6\}$, $\gamma_2 = \{0.45, 0.43\}$, $\gamma_3 = \{0.71, 1\}$, and $\gamma_4 = \{1, 0.33\}$. Fourth, n_2 determines the grey relational grades: $\Gamma_1 = 0.46$, $\Gamma_2 = 0.44$, $\Gamma_3 = 0.85$, and $\Gamma_4 = 0.66$. Finally, n_2 assigns content selection probabilities: $prob = \{0.19, 0.18, 0.35, 0.27\}$. Then, one content is selected based on these probabilities. Suppose that content c_3 is chosen. Accordingly, n_2 sends c_3's availability vector $\mathbf{a}_{n_2,3}$ to n_1. At time $t = t_1$, among c_3 candidate pieces to be transferred, n_1 selects the one with the lowest prevalence using an intra-content selection strategy (e.g., PACS) and sends it to n_2.

2.5. Summary

In this chapter, we reviewed some opportunistic content dissemination background and problem statement. We then presented EPICS, a generic and extensible distributed strategy based on the grey relational analysis for inter-content piece selection when two nodes observe a contact opportunity. The use of the grey relational analysis makes EPICS quite extensible: properly setting the reference data series x_0, we may give priority to contents having specific features like small/big size, early/late creation time, content type/thematic, etc. EPICS is designed to fairly and quickly exchange multiple and large contents in opportunistic networks.

Chapter 3

PePiT: An Android-based substrate for multi-content dissemination

Despite the number of opportunistic dissemination protocols and strategies based on users' mobility and social behavior, there are not as much real implementations. To fill the gap, we have developed an Android mobile application called PePiT[95] implementing the EPICS protocol (presented in Chapter 2). In the following, we present all the steps needed to bring EPICS from theory to practice as well as its evaluation in many scenarios.

3.1. Requirements

At high level, PePiT should be capable of:

- Detect other devices running the same application in the Wi-Fi range.
- Exchange with them a photo picked from the gallery or shot with the camera through Wi-Fi ad hoc.
- Receive and store photos sent by other devices.
- Notify the user when a new device joins the ad hoc network or when a device leaves it.
- Notify the user about the downloading progress.

If the requirements about the application itself are quite specific to the purpose of exchanging photos, we design the communication module in order to be as much general as possible and decoupled from the higher levels in order to be reused and called by other applications. Its features include: (i) start or join an ad hoc network, (ii) stop or leave an ad hoc network, (iii) broadcast messages, (iv) send messages to a specific neighbor, (v) operate with UDP and TCP sockets, (vi) operate with IPv6 addresses, and (vii) manage and pass received messages to any upper level applications.

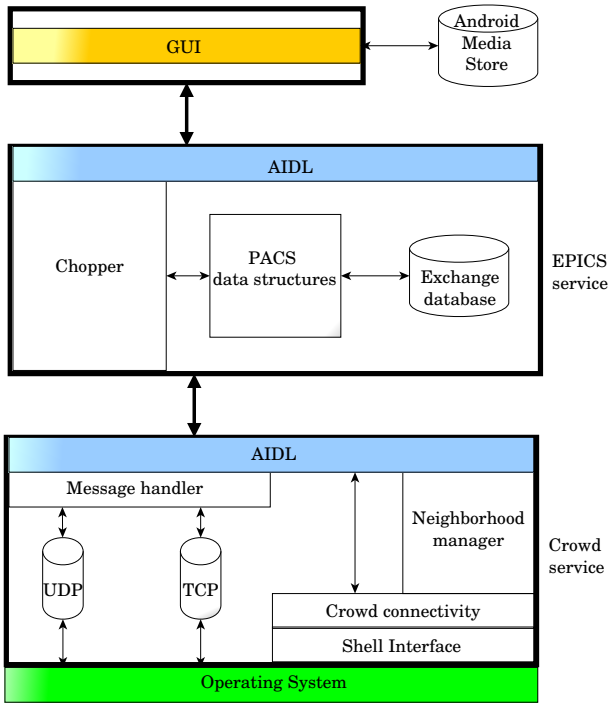


Figure 3.1: Modules of the PePiT architecture.

3.2. Architecture

To meet the requirements, we divided the system in three modules as depicted in Fig. 3.1.

Graphical User Interface. The application user interface provides the possibility to exchange a content (e.g., a picture) either by picking it out from the Android Media Store or by directly shooting it with the phone camera. It displays the received pictures, the downloading progress and all the network configurations. Users have also a view of the neighborhood, and they can monitor the wireless interface settings. The GUI is composed of two main activities (in Android jargon, an activity is what actually users see as window) and one abstract activity. Figure B.3 shows the UML class diagram related to these activities.

EPICS Component. Just below the GUI, this is the software module implementing the EPICS protocol. It is implemented as an Android remote service. This design approach gives two advantages. It allows EPICS to run on a different instance of the Dalvik Virtual Machine (the virtual machine where the Android code is executed) without interfering with the application itself that could require a large amount of heap memory. Secondly, any external application can bind this service and call the functionalities described in its Android Interface Definition Language (AIDL). In particular, the AIDL exhibits the following functions:

- *epicsPublish(String content_uri, int chunkSize)*: A function that enables application to publish a content using EPICS protocol.

- *epicsNotifyReceived(String content_uri)*: A callback function that is called whenever content is fully received using EPICS protocol.

The *chopper* sub-module cuts the content into K pieces filling the *availability vector* and creating the *prevalence vector* in the PACS data structure submodule. The EPICS service instantiates the EPICS protocol and runs in the background. It selects and sends content pieces using the CROWD service. Also, it tracks every received piece into an internal exchange database. For each received piece, it records a tuple containing: the unique content identifier, the sender's Android system identifier, the path where the content is locally stored, the piece number, the piece offset in bytes, and the piece size in bytes. When all the pieces belonging to the same picture are received, EPICS service rebuilds the picture and stores it in the Android MediaStore.

CROWD Component.[1] Even for the CROWD module we opted for a remote service design. It is composed of three sub-modules. The *Crowd connectivity* sub-module creates (or connects to) an ad hoc network. To do so, it relies on the ShellInterface utility class. This is an Android Operative System shell wrapper able to send system commands and return the output. Crowd Service also offers a *message handler* module which manages and tags along received and sending messages through TCP or UDP connections, in unicast or broadcast. The *neighborhood manager* broadcasts UDP beacons every T seconds in order to announce its presence to the neighbors. Beacons contain the IP address, the phone IMEI (International Mobile Equipment Identity), and the Android system identifier. The *neighborhood manager* also keeps state of neighboring devices: every time a beacon is received from an unknown device, it dynamically adds the new peer to the neighborhood list and shows up an Android notification to the user. The neighborhood list is internally scrolled every T_c seconds to check if some peers left the network. We denote $B_{x,n}$ the time of the latest beacon received from peer x, m the number of tolerated missed beacons and t the current checking time (multiple of T_c). If $t - B_{x,n} > T \times m$, the neighborhood manager considers that peer x left the network and shows a leaving notification to the user. Finally, the *message handler* sub-module creates and serializes outgoing CROWD messages and parses incoming ones.

Crowd service is accessible through an interface with the following methods:

- *received_msg(CrowdMessage msg, AssociateData ass)*. Called when a full Crowd message is received. The message is internally dissected in order to decode all the fields inside and to be consequently processed.
- *associate_join(AssociateData ass)*. Called to add a new neighbor.
- *associate_leave(AssociateData ass)*. If during the checking loop, a neighbor does not satisfy anymore the condition above, this API is called to remove it from the neighborhood.

[1] CROWD is the name of the ANR project supporting part of this work (http://anr-crowd.lip6.fr).

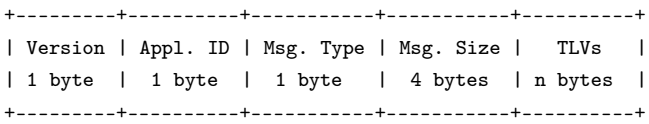


Figure 3.2: Crowd message format.

- *connected_to(AssociateData ass)*. A new TCP connection from a peer is received.
- *disconnected_from(AssociateData ass)*. Called when a TCP connection is lost.
- *broadcastMessage(CrowdMessage msg)*. Broadcast a CrowdMessage.
- *sendUnicast(CrowdMessage msg, AssociateData ass)*. Send a CrowdMessage to a neighbor through UDP.
- *sendReliableUnicast(CrowdMessage msg, AssociateData ass)*. Send a CrowdMessage to a neighbor through TCP.
- *connect_to(AssociateData ass)*. Establish a TCP connection.
- *disconnect_from(AssociateData ass)*. Close a TCP connection.
- *getAllAssociates()*. Get all the neighbors.

The UML class diagram of this module, instead, is showed in Figure B.1 of Appendix B.

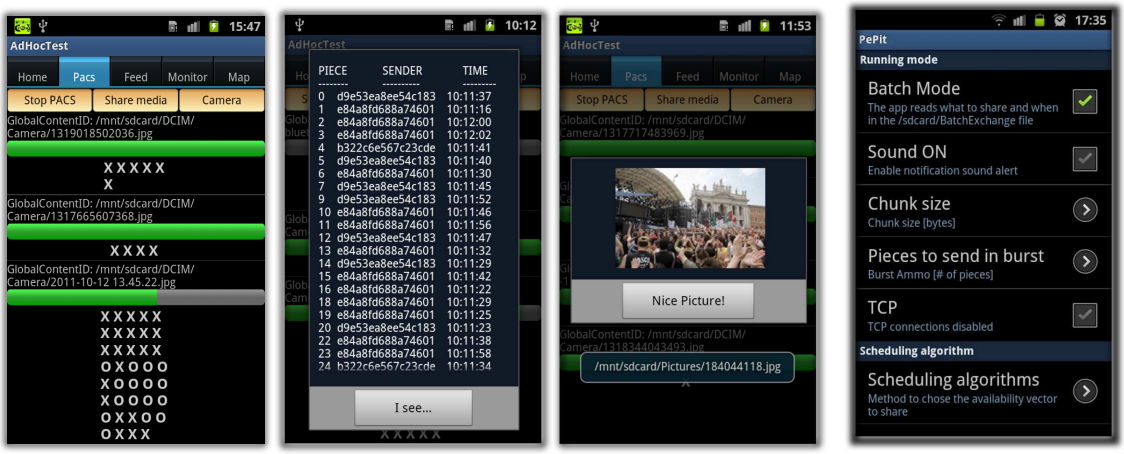
3.3. Internal data structures

EPICS component maintains at each node and for each content two data structures, namely availability and prevalence vectors. CROWD component manages the neighborhood, so it has a dedicated data structure to describe neighbors.

Messages are extensible and have the format presented in Fig. 3.2. The "`Version`" field indicates the version of the message encoding. As many applications can use component features, an "`Appl. ID`" field of one byte is reserved to encode the application identifier. The "`Msg. Size`" field contains the whole message size. The "`Msg. Type`" field is an application-dependent type of message. According to the "`Msg. Type`", an arbitrary number of TLVs (Type-Length-Value) fields may be included. Two types of messages are handled by PePiT: *availability vector message* and *data message*.

Availability vector message. A broadcast message. It contains four TLVs: the sender's Android identifier number, the unique content identifier, the associated availability vector, and the creation time.

Data message. A unicast message. It contains five TLVs: the sender's Android identifier number, the source's Android identifier number, the unique content identifier, the transmitted piece number, and the data. Each data message contains only one piece of content.



(a) PePiT. From left to right: exchanges in progress, history of content pieces received from different peers, preview of the received picture.

(b) PePiT settings menu.

Figure 3.3: PePiT user interface.

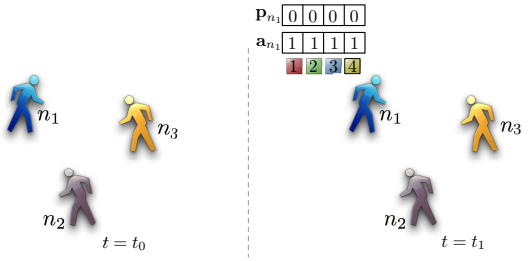
3.4. PePiT settings menu

PePiT has its own settings menu (Fig. 3.3(b)), where the user can easily change most of the parameters. If the batch mode is disabled, users can interactively take pictures from the gallery or using the camera and exchange them. If the batch mode option is enabled, the user selects a batch file stored on the SDCARD. The bench file plays the role of orchestrator. It contains, for each content, the creator's Android identifier number and the creation time. All the devices hold a copy of a common benchmark and generated contents based on it.

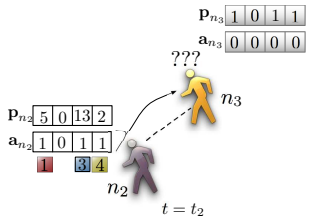
PePiT settings menu also contains other important customization settings which will be exploited in the next chapter; among others, we have the beaconing frequency, the chunk size, and the burst size.

3.5. Deployment on Android mobile devices

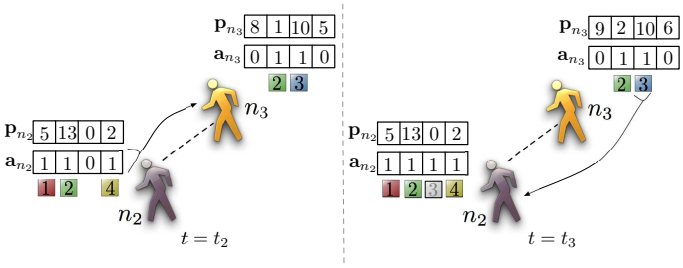
At implementation time, neither Wi-Fi Direct nor Bluetooth 4.0 APIs were available. We based our implementation on standard Wi-Fi capabilities then. This implementation choice however does not affect EPICS mechanisms. As the Android system does not provide an API to manage IEEE 802.11 ad hoc communications, it is foremost imperative to follow a procedure, called *rooting*, in order to gain administrative access rights on the phones. Thanks to this procedure, it is possible to run system-level commands just like in a Linux environment. Taking advantage of the Android NDK (Native Development Kit) facilities, Linux wireless tools for the ARM processor were compiled and wrapped into PePiT. These tools enable to create and connect phones to an ad hoc network. Listing B.1 shows a snippet of code to start an ad hoc network on Samsung Galaxy S II smartphones. Similar commands are provided for many other models.



(a) Creating content and initializing vectors.



(b) Case 1: user n_3 was not aware about the existence of content i.



(c) Case 2: user n_3 already got some piece of content i.

Figure 3.4: An illustrative scenario of the working of EPICS.

PePiT has been developed to run on mobile phones equipped with Android system with a minimum API level equal to 8 (Android Froyo, version 2.2.x). Fig. 3.3(a) shows some screenshots of the PePiT application. It has been successfully installed on a virtual machine (VM) running android-x86[1], a port of the Android system to x86 platforms. In this case, the host machine wireless interface is connected to the ad hoc network and bridged to the VM.

3.6. An illustrative scenario

To illustrate the operation of PePiT in a real deployment, we present an illustrative scenario with three users (Fig. 3.4). At time t_0, no content is available. Hence, no user maintains any availability or prevalence vectors. At time t_1, user n_1 creates a content. Then, *epicsPublish* function is called. Based on the user ID of n_1 and on a local identifier, the function assigns a unique global identifier i to the content. *epicsPublish* function also chops the content and initializes associated vectors (for the sake of illustration, suppose that the content is chopped into four pieces). An availability vector $\mathbf{a}_{n_1,i}$ is created with all bits set to 1. Also, an empty prevalence vector $\mathbf{p}_{n_1,i}$ is generated (Fig. 3.4(a)). Periodically, each user selects an availability vector and broadcasts it. The selection of the availability vector is done based on GRA as depicted in Section 2.4. Let us assume that the availability vector $\mathbf{a}_{n_2,i}$ associated to content i is selected and transmitted from user n_2 to user n_3 at time t_2. This could only happen if user n_3 is neither transmitting nor receiving data. Two cases are possible:

Case 1: user n_3 was not aware about the existence of content i (user n_3 has no vectors associated to the content i). Then user n_3 creates an availability vector $\mathbf{a}_{n_3,i}$ with all bits set to 0 and a prevalence vector $\mathbf{p}_{n_3,i}$ with all values set to 0 except the one associated to the pieces that user n_2 holds that are set to 1 (Fig. 3.4(b)).

Case 2: user n_3 already got some pieces of content i (user n_3 has vectors associated to the content i). In this case, user n_3 updates its prevalence vector based on the availability vector of node n_2. If user n_3 holds some pieces of content that user n_2 does not, user n_3 selects the less prevalent one and sends it to user n_2 (Fig. 3.4(c)). At time t_3, user n_2 receives a piece of content from user n_3. User n_2 stores the piece in its local memory and updates the availability vector $\mathbf{a}_{n_2,i}$ associated to the content i. It sets to 1 the bit associated to the piece just received. If user n_2 gets all the content pieces, *epicsNotifyReceived* function is called to notify the user that content has been fully downloaded.

3.7. Experimental setup

In this section, we summarize our experimental setup.

Table 3.1: Parameters for the first experiments with PACS and EPICS.

Parameters	*Values*
Transport protocol	UDP
Number of nodes	10
Number of contents	40
Piece size	64kB
Size of contents	16kB, 3.5MB
T_c	1 second
T	2 seconds
m	6

3.7.1. Experimental parameters

Transport protocol. Data pieces are sent using UDP datagrams.

Number of nodes. We placed 10 Android phones (4 HTC Desire and 6 Samsung Galaxy-S-II equipped with Android 2.3.3) on fixed locations in a 30m^2 office. Although we tested and validated the working of EPICS protocol in a mobile environment, we deliberately performed the experiment in a static configuration to be able to compare the results obtained using different strategies.

Number of contents. We consider the dissemination of 40 contents. Each device has four original contents stored on their SDCARD. Again, we used a batch mode in order to get exactly the same initial configuration (Section 3.4).

Piece size. Since pieces are sent through UDP, we set the piece size to fill the maximum UDP packet size (64kB). This way, each piece is transmitted in an unique UDP packet.

Size of contents. Even if a content may be smaller than the piece size defined above, still a CROWD data message must be created and sent through an UDP packet. Thus, we consider that content sizes vary from 1 piece (16kB) to 56 pieces (3.5MB).

Content creation times. Contents are created at different times of the experiment after a warmup period of 120 seconds to be sure that all nodes have initialized the internal structures. For complete automation and to make all the applications start at the same time, we developed an NTP client application to synchronize the smartphone internal clock to an NTP server and we used TaskBomb[108], an application which acts as Unix's Cron utility for Android.

Beaconing parameters. As described in Section 3.2, the neighborhood list is periodically scrolled. For the neighborhood management, we set $T_c = 1$ s, $T = 2$ s, and $m = 6$.

Availability vector message parameters. We made broadcast an availability vector message at the same frequency of beacon messages.

Experiment parameters are summarized in Table 3.1.

3.7.2. Discussion

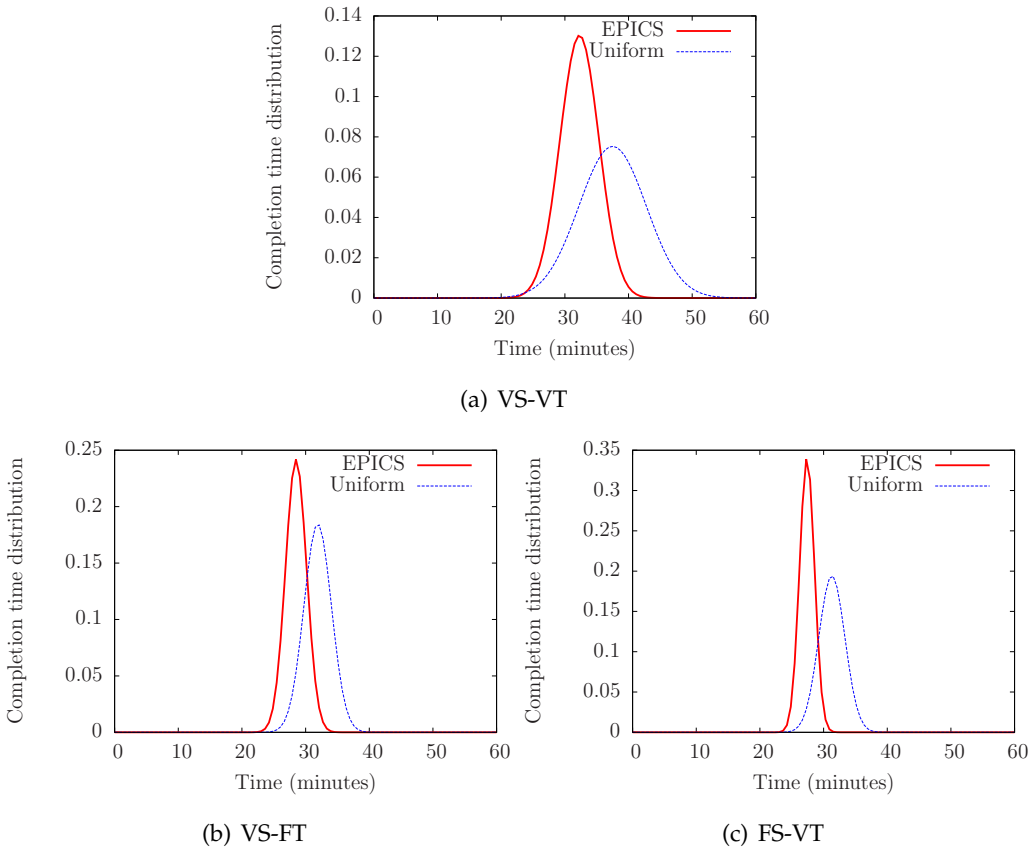
Beaconing parameters set above mean that, unless moved, a node is seen as disconnected and deleted from the other nodes' neighborhood list, only if six of its beacons are missed. Tuning these parameters we can emulate nodes contacts and inter-contacts. Many other combinations are possible: if we want to emulate mobile nodes with large inter-contact times, and we want to save open TCP connection, we can increase the tolerance factor m. On the other hand, if we want to make appear and disappear a node from the others' neighborhood list, then we increase its beaconing period T, keeping low T_c and m values for the others. Although we tested and validated the working of EPICS protocol in a mobile environment, we first performed the experiment in a static configuration to be able to compare the results obtained using different strategies.

3.7.3. Benchmarking

We compare EPICS to the uniform strategy. We call the uniform strategy, a strategy that use PACS for intra-content selection and that selects the content to transmit in a random uniform way. The experimentation details are roughly the same. There are two main differences though: *(i)* In order to investigate the impact of the creation time, contents are created at longer time ranges. Creation times vary between 0 seconds and 1,030 seconds after the warmup period. *(ii)* The experiments last as long as required for both EPICS and uniform strategies to achieve full dissemination of all contents.

3.8. Experimental results

We compare EPICS to the uniform strategy. We call the uniform strategy, a strategy that, as EPICS, uses PACS for intra-content selection, but selects the content to transmit in



(a) VS-VT

(b) VS-FT

(c) FS-VT

Figure 3.5: Distribution of completion times for experiments.

a random uniform way. Experiments last as long as required for both EPICS and uniform strategy to achieve full dissemination of all contents.

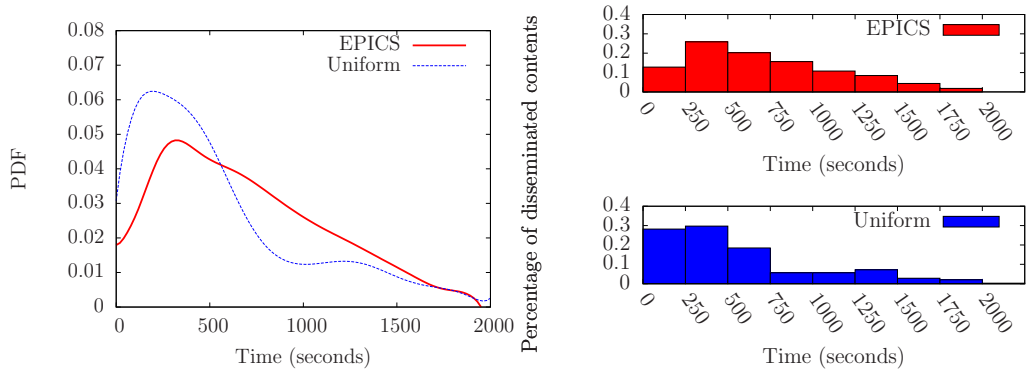
We test three scenarios:

- Variable size and variable creation time **(VS-VT)**. Forty contents (ten per node) with sizes from 16 (1 piece) to 3.5 MB (56 pieces) are created at different moments.
- Variable size and fixed creation time **(VS-FT)**. The same contents are created simultaneously after a warmup period of 2 minutes.
- Fized size and variable creation time **(FS-VT)**. Forty contents of 140 kB (3 pieces), ten per node, are created at different instants.

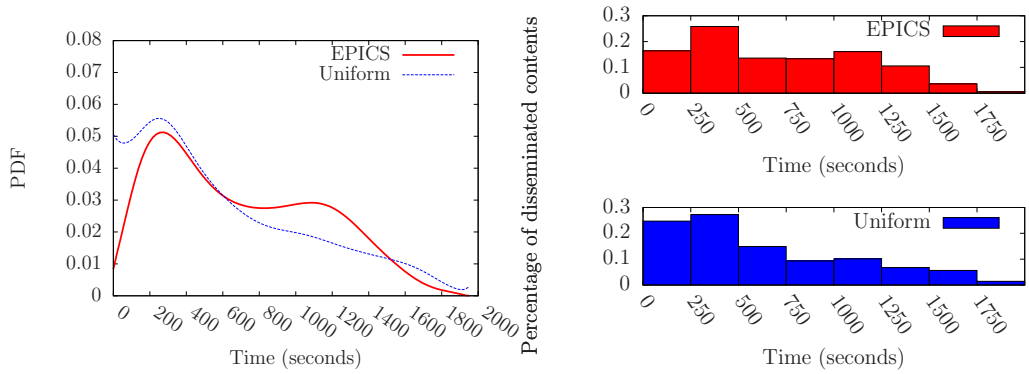
We repeat each scenario ten times and we get the average complete diffusion μ time and standard deviation σ. With μ and σ we build the normal distributions shown in Figures 3.5. Not only EPICS is faster than the uniform, but it also has a narrower variance meaning that it tries to complete a fair dissemination among all contents regarding the size and the creation time. In particular we have these values (in minutes):

- VS-VT. Uniform [$\mu = 37.5, \sigma = 5.3$], EPICS [$\mu = 32.3, \sigma = 3.06$].
- VS-FT. Uniform [$\mu = 31.9, \sigma = 2.16$], EPICS [$\mu = 28.5, \sigma = 1.65$].

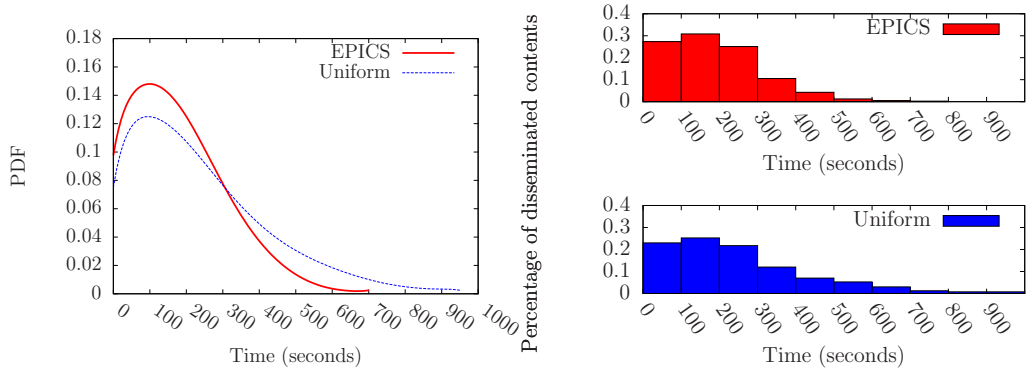
- FS-VT. Uniform [$\mu = 31.3$, $\sigma = 2.06$], EPICS [$\mu = 27.4$, $\sigma = 1.17$].



(a) VS-VT



(b) VS-FT



(c) FS-VT

Figure 3.6: Content dissemination delays.

Dissemination delay distribution. We now want to figure out if the faster dissemination evolution obtained with EPICS is induced by more homogeneous content dissemination delays. We investigate the distribution of content dissemination delays when either content size or creation time is fixed (Fig. 3.6(b) and 3.6(c), respectively) and when both size and creation time vary (Fig. 3.6(a)). In the scenarios VS-VT and VS-FT, even if the Uniform presents more contents disseminated in the first 600 and 400 seconds respectively, the latest contents spend much more time to be fully disseminated. Conversely, EPICS obtains fairer

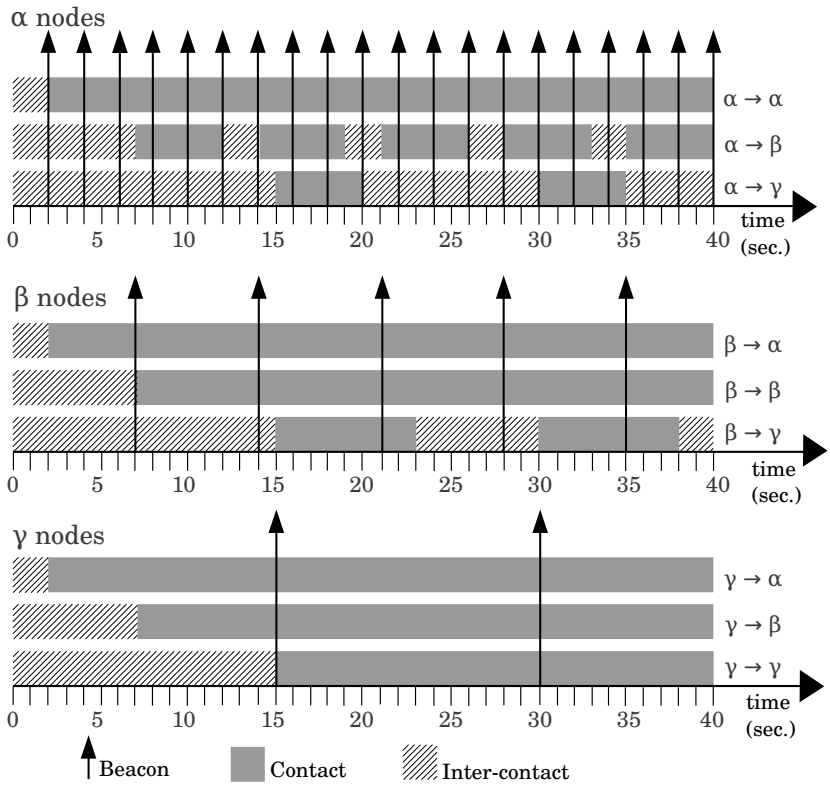


Figure 3.7: Contacts and inter-contacts in the emulated mobile scenario. All nodes start at the same time.

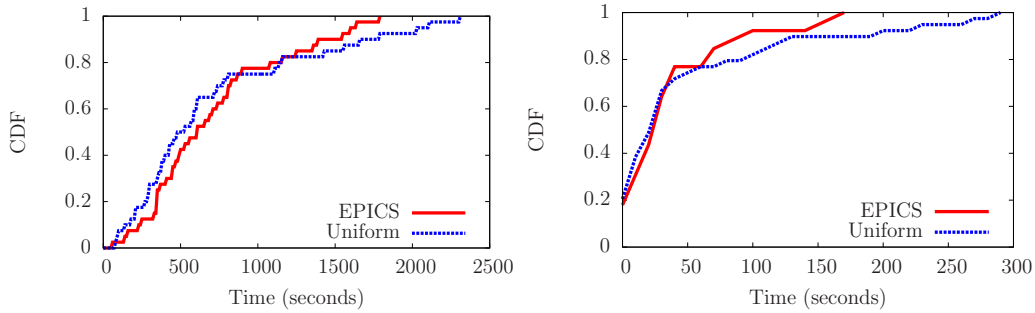
Table 3.2: Parameters for nodes of group α, β and γ.

Group	T (sec.)	T_c (sec.)	m
α	2	1	2
β	7	1	1
γ	15	1	1

dissemination delays by mitigating the skewness and reducing outliers. In the case FS-VT, with EPICS, dissemination delays are concentrated in the first 600 seconds, while with the uniform strategy they reach 1,000 seconds.

3.8.1. EPICS in an emulated mobile scenario

We test EPICS performance versus the Uniform strategy in an emulated mobile scenario. For this kind of evaluation, we actually care about contacts and inter-contacts among nodes and not the mobility itself. This is the reason why we emulate the mobility. The scenario is created dividing nodes in three groups (α, β, γ) and tuning the beaconing parameters as explained in Section 3.7.2. Table 3.2 summarizes these parameters for each group, while Figure 3.7 shows the evolution of contacts and inter-contacts among groups. Three nodes



(a) Cumulative distribution function of dissemination times.

(b) Cumulative distribution function of time gaps between consecutive disseminations.

Figure 3.8: EPICS versus Uniform in an emulated mobile scenario. Contents have variable size and variable creation times.

constitute the γ group. In this scenario, they are always in contact with all the others, since the first beacon they receive, as they consider a node out of their neighborhood only if they do not receive a beacon from it within 16 seconds. On the other hand they send beacons with a low frequency: every 15 seconds. For this reason, α and β nodes are often in inter-contact with γ nodes. We have set other three nodes in according to the β group parameters. These nodes are always in contact with α and β nodes, but they spend almost the same time for contact and inter-contact with γ nodes. α nodes are always in the neighborhood of the other nodes as they send a beacon every 2 seconds. On the other hand they have a low tolerance to missed or delayed beacons from other nodes. Most of the time they are disconnected to γ nodes and they present inter-contact gaps also for β nodes. γ nodes can be thought as devices having a wider contact range, α node at the contrary.

In this scenario we disseminate the same 40 contents having variable size and variable creation time of Section 3.8. Figure 3.8(a) shows the content dissemination evolution during time. Similarly to the stationary case, also in this case the Uniform strategy is slightly faster disseminating the first 75% of the contents (many of these are one piece content, while EPICS tries to start with larger contents first), then it slows down. At the end EPICS completes the dissemination exactly seven minutes faster.

In Figure 3.8 we compare the distribution of temporal gaps between consecutive diffusion delays. For both strategies, 75% of disseminations are spaced in time of one minute maximum. Nevertheless the maximum gap is 150 seconds for EPICS and almost the double for the Uniform.

3.8.2. EPICS in a mobile scenario

In Section 3.8.1 we have shown how to test EPICS in an emulated mobile scenario taking advantage of PePiT features. In this section we compare EPICS versus the Uniform strategy in a real mobile scenario. We use six smartphones, four of them move along the path

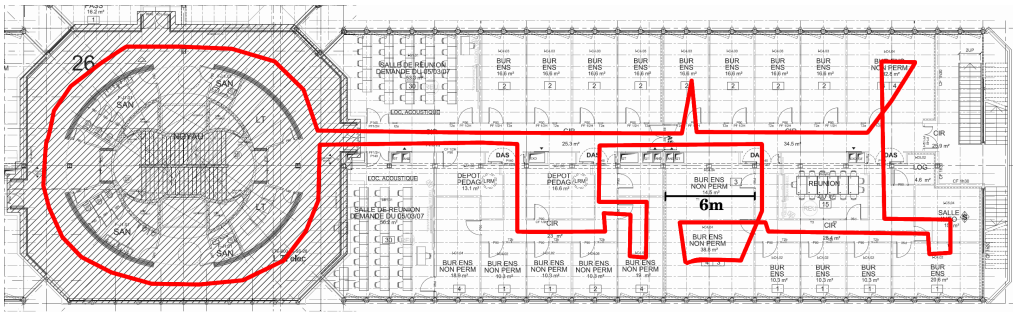
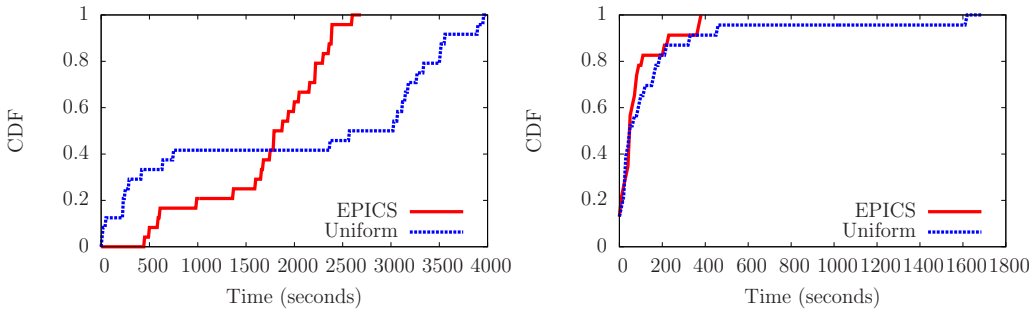


Figure 3.9: Mobile scenario plan. Path is highlighted in red. All nodes start at the same time.



(a) Cumulative distribution function of dissemination times.

(b) Cumulative distribution function of time gaps between consecutive disseminations.

Figure 3.10: EPICS versus Uniform in a mobile scenario. Contents have variable size and variable creation times.

highlighted in Figure 3.9 at the first floor of the LIP6-UPMC laboratory in Paris, while last two have a fixed location along the path. Each smartphone generates four contents having variable sizes at different instants.

Results shown in Figure 3.10 confirm the dissemination evolution presented in the other scenarios. The Uniform strategy performs well at the beginning disseminating one piece contents, but, on the long period, EPICS is almost double quicker (Figure 3.10(a)). The time gap between two consecutive dissemination delays can achieve 20 minutes with the Uniform, while a maximum of 5 with EPICS (Figure 3.10(b)).

3.9. Summary

Opportunistic content sharing among mobile users is expected to be a widespread application in a near future, as collocated people are likely to share mutual interests. In this Chapter we introduced PePiT an Android application with the goal to fill the lack of real opportunistic content dissemination protocol implementations. We retrace the developing process from the requirements to the complete implementation.

With PePiT we can evaluate the performance of EPICS in many scenarios. Our testbed is composed by 10 Android smartphones nodes. In according to the experimental scenario, nodes are placed in fixed positions always in contact or they emulate contacts or they are

mobile. We compare EPICS against a uniform strategy. EPICS ensures fairer dissemination delays for all the contents regardless of their creation times and sizes.

PePiT reveals itself to be very extensible. Keeping the lower opportunistic communication module, it can be used for any upper layer protocol. As a side aspect, in order to make PePiT available to the general public, practical questions should be addressed, as the reset time of the prevalence vector. As a future work we wish to make PePiT runnable also on non-rooted stock smartphones.

Chapter 4

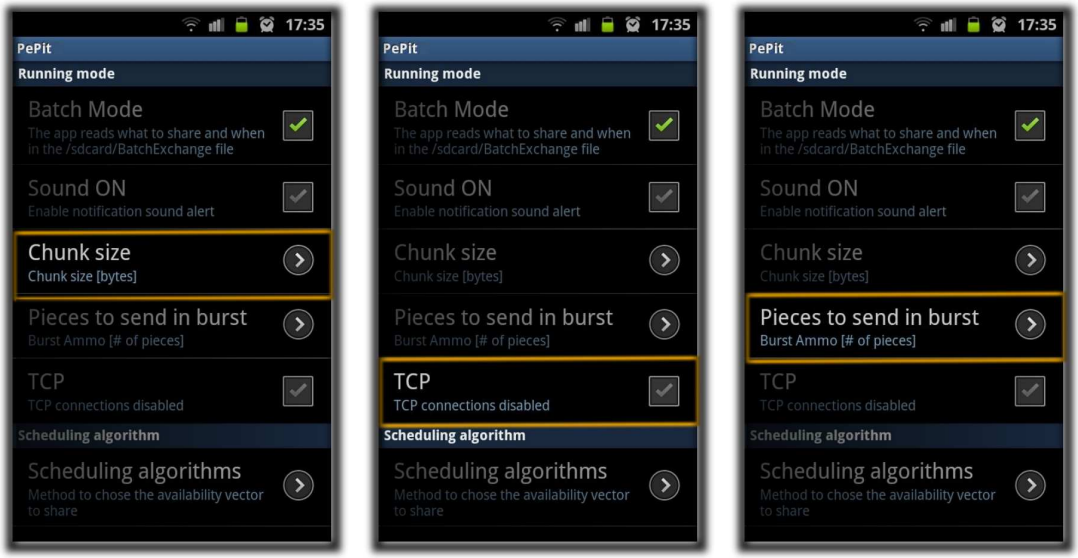
DAD: Bringing dynamics to EPICS

In Chapter 2 we described EPICS, an inter-content selection strategy to exchange large contents in an opportunistic way. The implementation of EPICS on PePiT gave us the possibility to prove and evaluate its performance versus other solutions with real world devices. In this chapter we answer to the question: is there a way to further improve EPICS? To this end, we experimentally observe the impact of some protocol parameters tunings and we finally propose DAD (Dynamically Adaptive Dissemination), our solution to reduce the generated overhead adapting to the dynamic neighborhood.

The chapter is organized as follows. In Section 4.1, we describe the rationale behind our solution and, in Section 4.2, we present our experimentation campaign. The experimental results are reported in Section 4.3 – they show that there is indeed room for improvement. We propose then DAD, a solution that extends EPICS and makes it react to varying network conditions to better exploit contact opportunities.

4.1. Rationale

When nodes get in contact and want to share some contents using EPICS, firstly they exchange one of their availability vectors (of one of the contents) and update their respective prevalence vector. This preliminary stage has the goal of maximizing the utility of the contact by choosing the right piece to transmit. Nevertheless, this expedient does not fully avoid duplicated pieces since the decision about which piece to send is independently taken by each of the neighbors that happen to be in contact with the node at the same time. Also, such a strategy can result in non negligible overhead, as the transmission of each piece is preceded by the transmission of an availability vector. One question seems appropriate here: *would it be interesting to send a burst of pieces, among the less prevalent ones, at each exchange of an availability vector?* To this end, would it be worth opening a reliable connection to exchange several pieces at once instead of using UDP connections, and modulate the piece size to fit the burst into a contact window?



(a) Piece size option. (b) Transport level protocol option. (c) Burst size option.

Figure 4.1: Screenshot of PePiT option settings.

To address these questions, we proceed *ad explorandum*: we tweak several parameters to find out which one has the most significant impact on the performance of EPICS in terms of dissemination latency. Like a pipeline, at each stage we set up a befitting experiment to find the best values for the parameters we consider; this configuration serves as the input for the subsequent experimental stage. We explore the influence of the piece size, the choice of the network transport protocol, and the impact of the size of the burst (see Figure 4.1).

During each experiment, we capture the wireless traffic using a passive monitoring system. Wireless traces will support us to better understand the application behavior and some unexpected results.[1]

4.2. EPICS breakdown

We conduct a campaign of experiments to check the best values for parameters as piece size, transport level network protocol, and burst size. To be as fair as possible, for the measurements reported in this chapter, we used smartphones with exactly the same hardware: eight Samsung Galaxy S II with a Dual-core 1.2 GHz Cortex-A9 CPU and 1 Gbyte RAM. Wireless capabilities (Bluetooth and WLAN) are managed by the Broadcom BCM4330 chipset. WLAN features include IEEE 802.11b/g/n standards, possibility to operate in the range of frequency [2.4–2.497] and [4.9–5.85] GHz, and several modulation techniques (OFDM, CCK, DQPSK, and DBPSK).

[1] As stated previously, wireless measurement can be so useful that we decided to focus on this aspect in Part II of this thesis.

Smartphones are laid down in the middle of a desk, with two monitors passively capturing traffic. No mobility is adopted to limit external interferences during this kind of experiments. At the beginning of each experiment, the clocks of the devices are synchronized. Every two seconds, nodes broadcast one of their availability vectors and stay tuned for pieces. At the beginning, only one node, the source, has contents to share. All the other nodes must wait to receive some chunks in order to act as a source too. In our experiments, contents are photos of exactly 3 Mbytes. We repeat the same experiment at least three times in different hours and days.

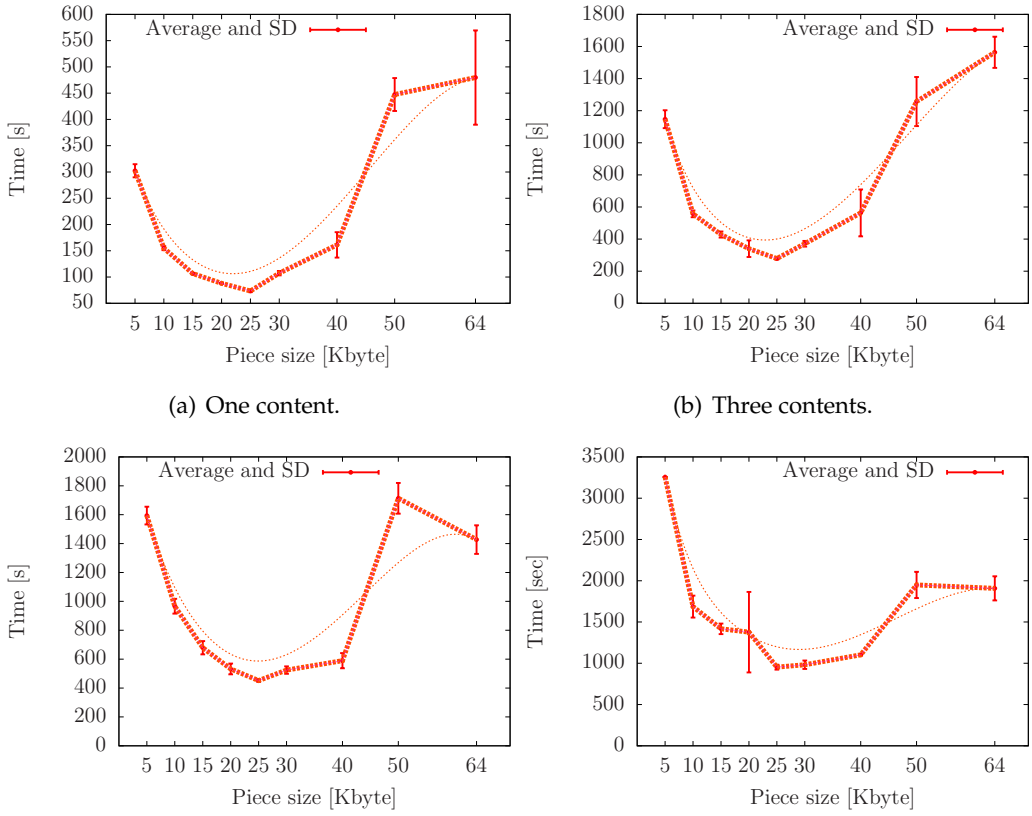
4.2.1. Impact of the piece size

The baseline protocols we consider (EPICS and PACS), as described in Section 2.3, get inspiration from BitTorrent to chop contents into smaller pieces; for this reason, as a first step, we consider for the piece sizes the same values as considered in BitTorrent-related applications[38;59;67].

The overall number of pieces to be shared (over all contents) has significant impact on the dissemination time. Neglecting low level transmission times, the smaller the number of pieces to be transmitted, the faster the dissemination. In other words, for a given content size, the dissemination is quicker if the piece size is bigger. Then, starting from a piece size of 64 Kbytes (to fulfill a UDP packet), we gradually scale down the piece size approaching the MTU (1,472 bytes in our case).

Figure 4.2 shows the average dissemination time and related standard deviation needed to share one, three, five, and ten contents among seven nodes (plus one source). We choose the piece size in the set {64, 50, 40, 30, 25, 20, 15, 10, 5} Kbytes. In all cases, the diffusion time curve follows a parabolic-like shape: for very small or very large piece sizes the dissemination takes longer, while it decreases for medium values in the range of 15-40 Kbytes. In the case of very small values, the overhead (because of the transmission of availability vectors) plays a hefty role. In the case of very large pieces, instead, data transmission times and fragmentation take the lead.

Figure 4.3(a) shows the cumulative distribution function of times required by the transmitters and stored in the duration field of RTS frames. This time is composed of $RTS + SIFS + CTS + SIFS + DATA + SIFS + ACK$. *A fortiori ratione*, these times become larger and larger as the DATA itself is larger and therefore must be split in more fragments spaced out by as many $SIFS + ACK$ as necessary. All the devices hearing a RTS followed by a CTS must delay their requests to access the medium at least for the whole duration. In the case a piece has the maximum size, 70% of RTS have a duration greater or equal to 1,512 μseconds. The values are 40% and 20% for piece sizes of 25 Kbytes and 15 Kbytes, respectively. If the piece is of 5 Kbytes, all the durations required are less than 1 millisecond. *A latere*, a bigger



(a) One content.

(b) Three contents.

(c) Five contents.

(d) Ten contents.

Figure 4.2: Dissemination time by tuning the piece size and using UDP sockets.

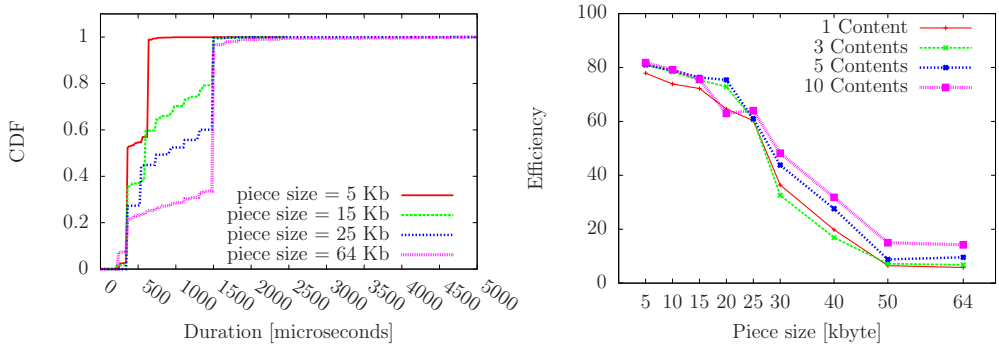
piece size also means more fragmentation and higher probability to lose a fragment (and forced to send it again).

Figure 4.3(b) shows the delivery efficiency calculated as the ratio of the amount of pieces put in the transmission queue over the number of pieces that must be received to complete the dissemination. For any number of contents to share, curves have the same shape: for small pieces (5, 10, 15, or 20 Kbytes), the efficiency is around 70%, while for bigger pieces (50 or 64 Kbytes) the efficiency is very low, around 10%. A significant slope starts after 25 Kbytes. This means that, for bigger pieces, we are forced to send, in proportion, more pieces than the one we effectively need to complete the diffusion.

It happens that there is a tradeoff between sending a few large packets or several small packets (generating more overhead); the right choice is in the middle, with a piece size of about 25 Kbytes.

4.2.2. Impact of the transport layer protocol

UDP is considered as the best choice to share contents in opportunistic networks since it does not need to create a stateful connection. We evaluate this assumption proposing the same experiment setup as before, but, this time, every time a node discovers a neighbor, it



(a) CDF of duration times specified in RTS frames for 10 contents.

(b) Delivery efficiency.

Figure 4.3: Data-link and application level measurements tuning the piece size and using UDP sockets.

opens a TCP connection with it. Thus, pieces are reliably sent while beacons and availability vectors are still sent through UDP. With these experiments, the goal is to consolidate what we figured out about the piece size in the previous section.

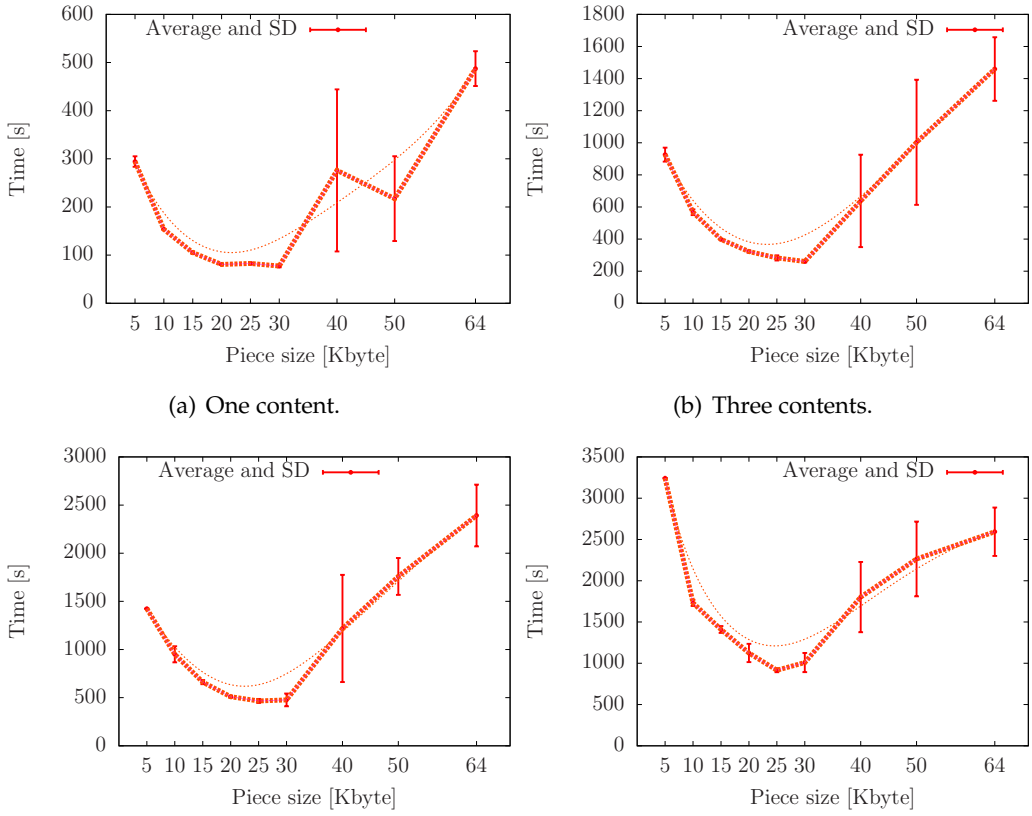
Figure 4.4 presents the average diffusion time to share one, three, five, or ten contents with seven nodes (plus one source) changing the piece size in the set {64, 50, 40, 30, 25, 20, 15, 10, 5} Kbytes. In this case too, for any amount of content to share, we detect a parabolic-like shape: for larger or smaller pieces, we experience longer dissemination times, and shorter times for middle range piece sizes. Starting from a piece size of 30 Kbytes up, the standard deviation becomes larger for at least five experiments. It witnesses the effort to send back again big packets after a loss.

Figure 4.5 shows the average diffusion time difference between the usage of UDP and TCP. In the case of one or three contents to share (Figures 4.5(a) and 4.5(b)), UDP seems to perform slightly worse than TCP, especially when the piece size is bigger than 40 Kbytes. Anyway, for five to ten contents (Figures 4.5(c) and 4.5(d)), results suggest the opposite behavior. But when TCP performs worse, it takes much longer than UDP. We conclude that UDP is the best choice and, also in this case, the best piece size in terms of efficiency is around 25-30 Kbytes. Finally, as shown in Figure 4.6, the smallest negative slope is at 25 Kbytes; thus, we consider in the following this piece size, unless specified.

4.2.3. Impact of the burst size

Once the piece size tuned and the transport protocol established, we need now to investigate if it is worth sending more than one piece at each exchange of availability vector. With the same experimental setting of previous sections, we exchange five contents, modulating the burst size in the set of {2,3,5,10} pieces and comparing the dissemination performance with no burst at all (only one piece exchanged per each exchange of availability vector).

Performance, in terms of dissemination time, becomes worse and worse as the burst size increases. Figure 4.7(a) shows, for each burst size, the elapsed time to achieve a complete



(a) One content. (b) Three contents.

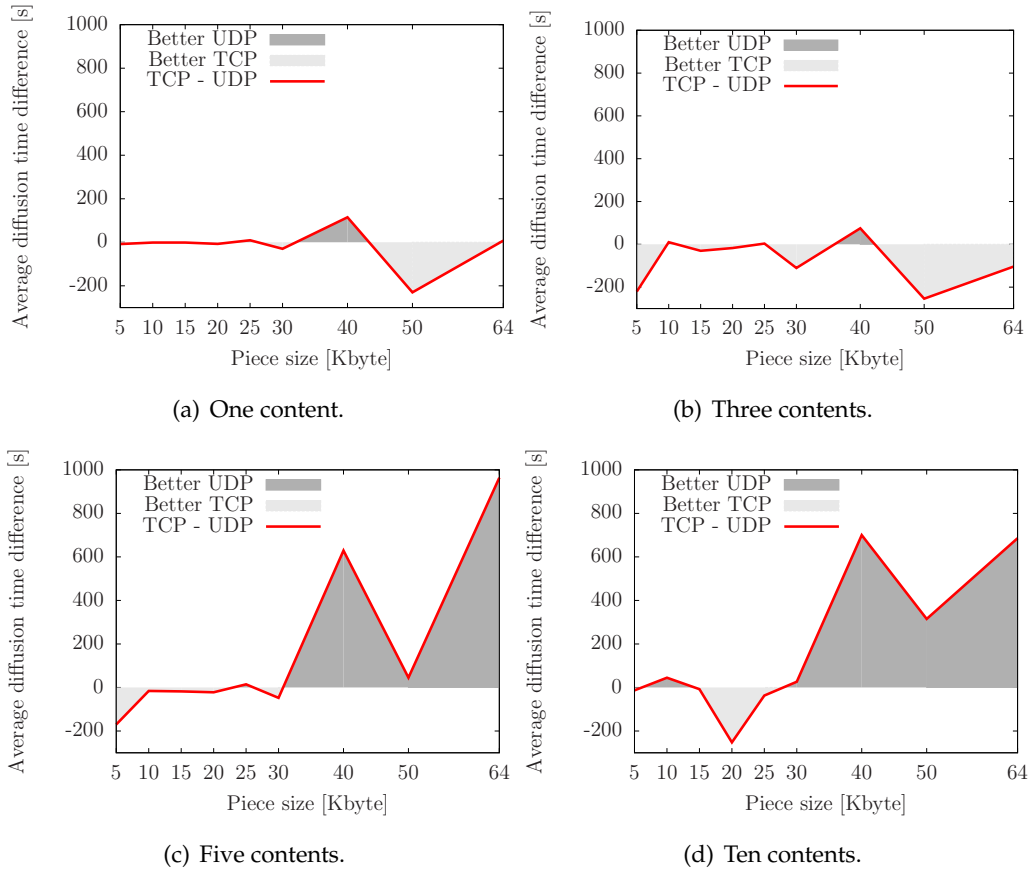
(c) Five contents. (d) Ten contents.

Figure 4.4: Dissemination time tuning the piece size and using TCP sockets.

dissemination of the five contents on one node, two nodes, until all the seven nodes requiring the contents. In many cases, with a $burstsize = n$ the dissemination on the seven nodes is faster than the dissemination on only one node in the case of $burstsize = n + x$. These results are completely independent from the external wireless traffic listened (Figure 4.7(b)). Even if in the case of $burstsize = 3$ we recorded less external traffic than in the case we do not use any burst, it takes exactly three times more to disseminate the contents.

Let us suppose a $burstsize = 10$ as shown in Figure 4.7. At each contact with another node, at most ten packets are placed in the transmission queue. The queue grows ten times faster than the basic solution without burst. These ten packets correspond to ten pieces the other node does not have, and chosen among the less prevalent ones. They are chosen based on a local and contemporary view. For each packet, a node must gain access to the medium waiting to be idle or reserving a slot with the $RTS - CTS$ mechanism. In this way, when packets in the tail of the queue (e.g., pieces to node 3 in the figure) eventually reach the head, they are likely to be obsolete, wasting transmission slots (i.e., other neighbors may have already sent it to the node, as shown in Figure 4.7).

We investigate if this phenomenon occurs varying the quantity of nodes involved in the content exchange. We share one content of 3 Mbytes with only one source and one node requiring the content (two nodes in total), one source and two other nodes (three nodes),



(a) One content.

(b) Three contents.

(c) Five contents.

(d) Ten contents.

Figure 4.5: Average dissemination difference using TCP and UDP sockets.

until eight nodes in total, and tuning the burst size from one to ten. Figure 4.9 shows that dissemination is faster using a larger burst when there are only few nodes, while, with six or more nodes in contact, dissemination is faster disabling the burst (or, the same, using a $burstsize = 1$). In particular, in the case of two nodes (including the source), is to be avoided a burst in the range $1 - 2$, while, increasing the burst in the range $4 - 10$, the dissemination time is divided by five, with a minimum at $burstsize = 10$. In the case of three nodes, the minimum values of dissemination time are expected with medium values of burst. In the case of four nodes, the curve starts to rotate: the best values are around a $burstsize = 3$. From six nodes, values of burst greater than one lead to a slower dissemination.

4.3. DAD: Dynamically Adaptive Dissemination

4.3.1. Room for improvement

We showed in Section 4.2.3 how tuning the burst size can either improve or worsen the dissemination performance based on the number of nodes in contact with the source. We collapse Figures 4.9(a)–4.9(g) into Figure 4.10, where the red line connects the burst size values in order to have the minimum dissemination time based on the number of nodes in

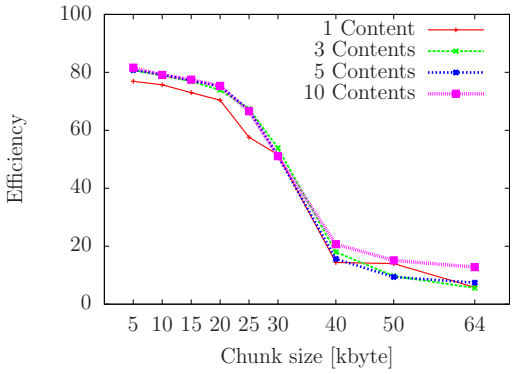
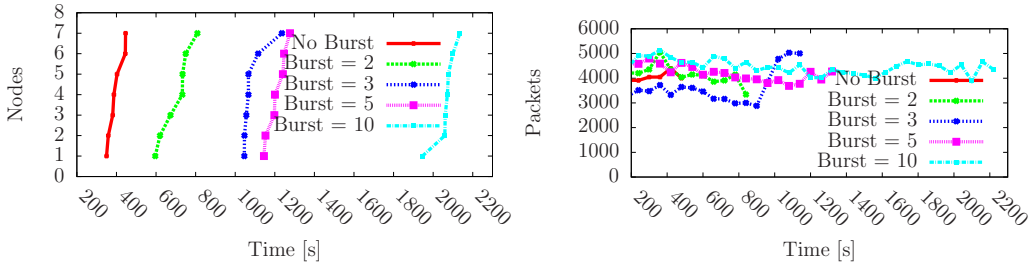


Figure 4.6: Dissemination efficiency tuning the piece size and using TCP sockets.



(a) Completion time per node.

(b) External traffic during experiments.

Figure 4.7: Dissemination time and external traffic with burst mode activated.

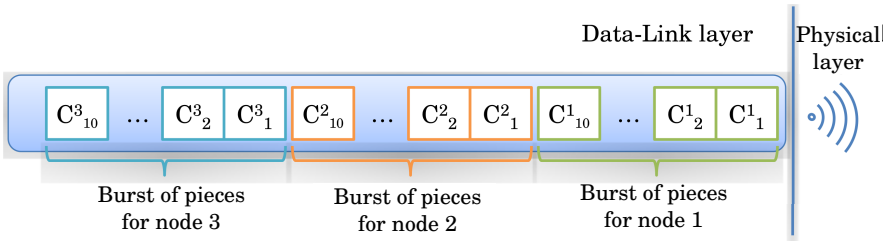


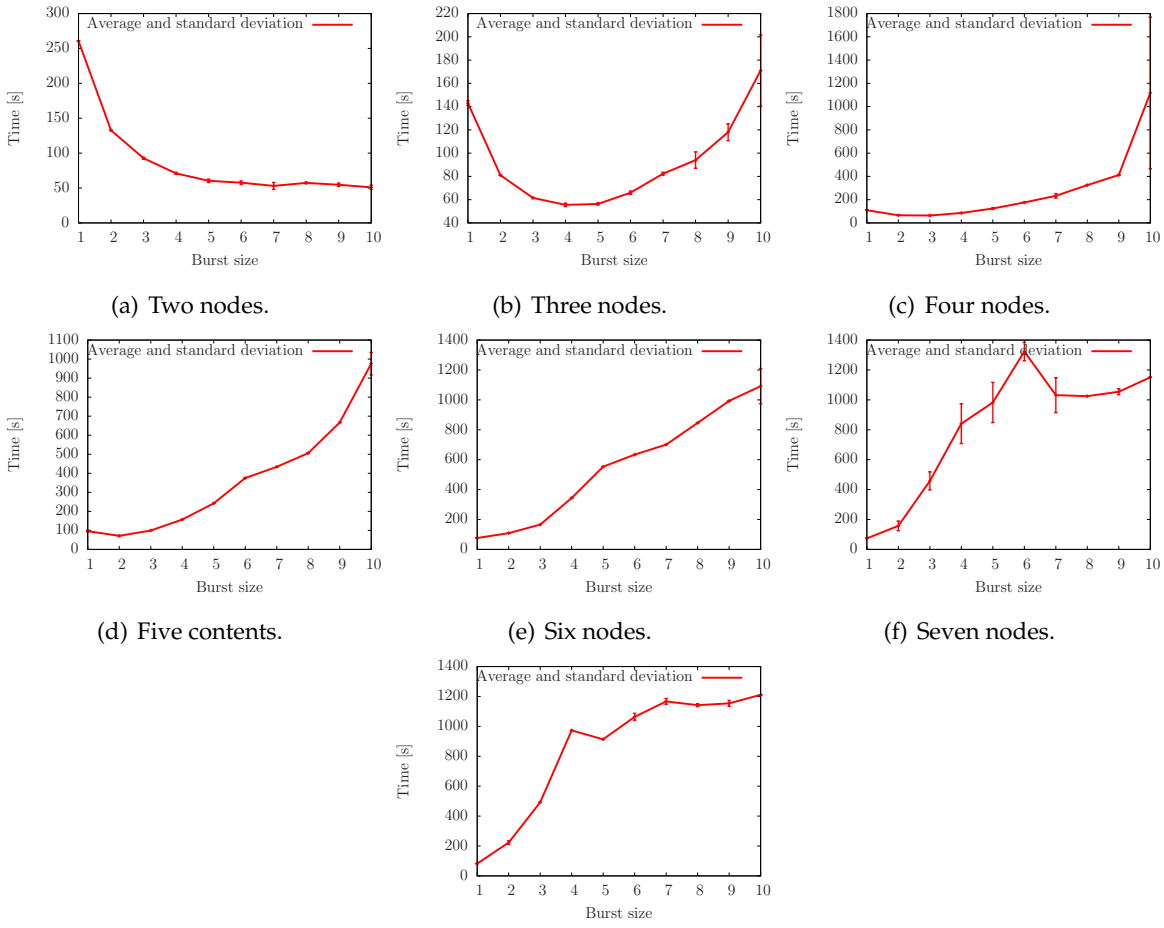
Figure 4.8: View of the transmission queue. With a burst size = 10, pieces experience a long queue delay and they result useless once transmitted.

contact with the source node. The gray area includes values of burst for a diffusion time at most 30 seconds after the minimum. Note that this area becomes narrower and narrower as the number of nodes in contact grows. In this plot, EPICS moves on the bottom, meaning that it can be improved up to a node degree of four. From a node degree of five up, it is worth sharing only one piece per exchange of availability vector.

To check how important this profit margin is, we analyze in the following some real and synthetic mobility traces. In particular, we examine the cumulative distribution function of node degree at every beaconing instant. We exclude from the distribution isolated nodes, as they are cannot exchange content with anyone.

We consider the following mobility traces:

Shopping MAll[35]. This is a dataset of real-world Bluetooth contact data collected from a



(a) Two nodes. (b) Three nodes. (c) Four nodes.

(d) Five contents. (e) Six nodes. (f) Seven nodes.

(g) Eight nodes.

Figure 4.9: Dissemination time of one content, tuning the burst size and changing the number of nodes involved.

mall in Nottingham (UK). For six days (following shops opening time), 25 devices captured Bluetooth contacts.

KAIST[89]. Another real-world dataset, consisting of 92 daily GPS track logs collected from the KAIST university campus in Daejeon, South Korea. Traces have been overlapped in time to produce one single trace. We assume a contact range of 10 meters as long as the Bluetooth trace.

SIMPS synthetic traces. We developed a mobility simulator, shown in Figure 4.12, based on a mobility model of human crowds with pedestrian motion called SIMPS[18]. We simulated a relatively dense toroidal space of 100 x 200 meters with 100 people moving for one hour. This model is based, among other parameters, on a "social radius". Nodes take decisions about their movements in according to the nodes they detect in that radius. In crowded environments, the social radius tends to shrink. Since we simulated a crowd model, we used a social radius of one, two, and three meters, varying the contact range accordingly as well.

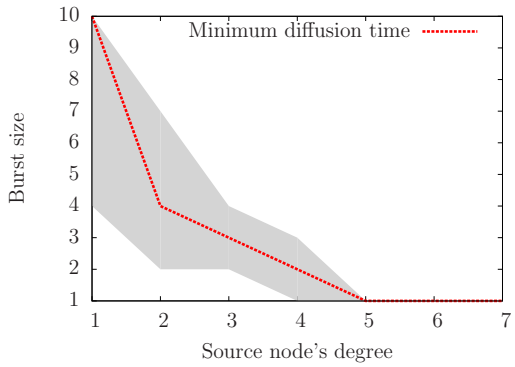
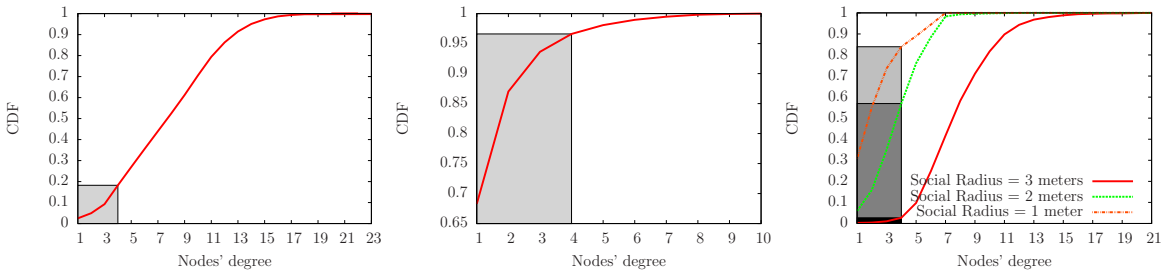


Figure 4.10: Operation area: Burst size for minimum diffusion time with a tolerance of 30 seconds.



(a) Shopping Mall. (b) KAIST. (c) SIMPS Simulator.

Figure 4.11: Cumulative distribution function of the number of nodes that every node perceives in his neighborhood, including itself. (a) Shopping Mall trace, (b) KAIST trace, (c) SIMPS trace.

Figure 4.11 shows the distribution of node degree for the traces we consider. The probability to find, at most, less than five nodes in contact considerably varies from trace to trace. The Shopping Mall in Nottingham has a surface area of 10,880 square meters (without considering the parking area). Being a mall, we can image it very crowded, especially during rush hours. Anyway, not everyone has a device with Bluetooth enabled. In these conditions, we can detect nodes in contact with, at most, other 4 nodes, with a 20% probability.

The KAIST campus is 1,432,882 square-meter wide. This huge size makes it possible to exhibit a very high probability (more than 95%) of, at most, less than five nodes in contact. For the sake of fairness, being a GPS trace, indoor places that should be the most crowded are not taken into account. In this case, using a large burst will largely improve the opportunistic exchange of content.

We also simulated a very dense scenario, with 100 people in a 100 x 200 meters plane. The improving margin considerably changes for slightly changes of social radius. In the case of three meters, there is only a 0.03% of improving margin. One more node helps achieve 0.1%. For a social radius of two and three meters, we get nodes with a degree of at most four with a probability of 55% and 85%, respectively.

We chose these traces because they are different in many aspects: nature (real and simulated), position (mall, university campus, simulated toroidal plane), log collection (Blue-

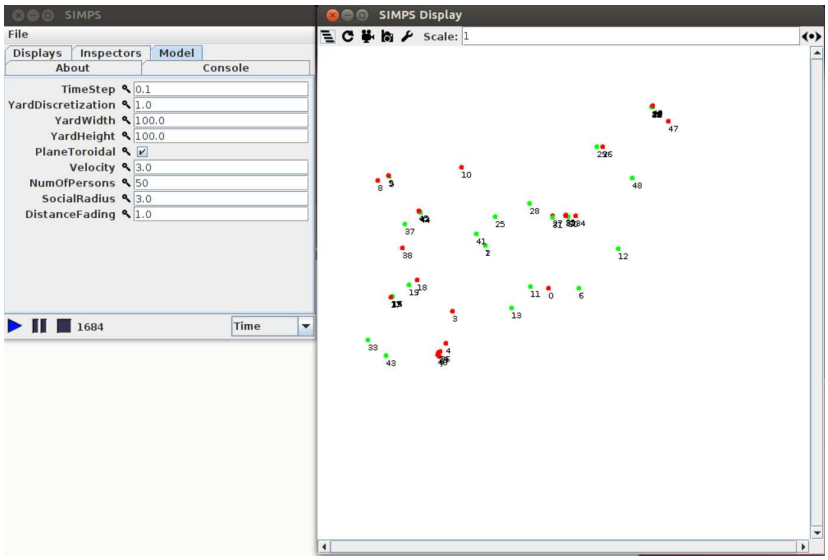


Figure 4.12: SIMPS simulator.

Table 4.1: Burst size in function of the node degree.

node degree	1	2	3	4	5+
burst size	10	4	3	2	1

tooth device, GPS, 2D position), plane size (medium, huge, small), density (medium, low, high). In any case, there is still a non negligible margin to improve opportunistic content exchange when nodes in contact are few.

4.3.2. DAD: Bringing dynamics to EPICS

We advocate including some flexibility in the the choice of the burst size. Based on the observations reported in the previous sections, we propose DAD (**D**ynamically **A**daptive **D**issemination), a solution that modulates the burst size according to the number of neighbors, always following the minimum diffusion time line of Figure 4.10.

DAD is a simple, yet highly efficient improvement of EPICS. At the time a node exchange an availability vector with a neighbor, this latter decides the number of pieces to include in the burst based on the degree of the node, as shown in the abacus depicted in Table 4.1.

We compare DAD versus the baseline EPICS version (i.e., with a $burstsize = 1$) and with an extreme case where EPICS sends bursts of ten pieces. To check the impact of the burst size, both DAD and EPICS have the same piece size (25 Kbytes). We start this experiment with only two nodes: one source that has ten contents of 3 Mbytes to share and another node. Then, every three minutes we add a new node requiring all the contents up to seven nodes. We gather relative completion times for each content from every node and we present the cumulative distribution function in Figure 4.13. Even if DAD and EPICS take the same time to diffuse all the contents to all the nodes, they present a considerable difference until the

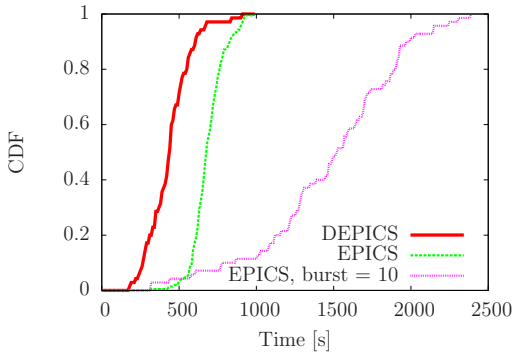


Figure 4.13: DAD vs. EPICS and vs. EPICS with a burst of ten pieces. CDF of dissemination times per content and per node.

97th percentile. It means that the dynamic adaptation not only facilitates the diffusion when there are only a few nodes, but, since the content is almost fully received in many nodes, these nodes can better support the dissemination even when we insert more nodes in the network. On the other hand, using EPICS with the maximum burst, the dissemination is very fast for the first few contents (i.e., when there are only few nodes), and then it slows down taking five times longer.

4.4. Summary

In this chapter, we deeply investigated the performance of EPICS implemented on top of PePiT. We tuned some parameters such as the piece size, the choice of the transport protocol, and the possibility to exchange a burst of pieces at each exchange of availability vectors. During our experiments, we captured the external wireless traffic and the one produced by our devices. Wireless traces allowed us to better understand some phenomena and results.

We conducted an experimental campaign consisting of more than 500 experiments, around four thousand application level logs and 60 Gbytes of wireless captured traces. From our experiments, we found out that the best configuration expects the exchange of 25 Kbytes chunks through UDP sockets. The burst plays a twofold role. A large burst size can support opportunistic content diffusion when the data exchange involves a few nodes (less than six in our experiments). With more nodes, it penalizes the exchange in terms of dissemination time. In this case, it is better to exchange only one chunk per contact, choosing it from a more updated set. As a general rule, then, it is worth to *send less, but send right!*

We developed a version of EPICS with a dynamic modulation of the burst size and we called it DAD. We showed that DAD can improve the content diffusion if it adopts correctly the observations we made previously.

Part II

WLAN monitoring: Basics, deployment, and collaborative behavior

Chapter 5

IEEE 802.11 traffic monitoring: Background and problem statement

Statistics from the second quarter of 2013 forecast that wireless LAN equipments and Wi-Fi phones market will exceed 7 billions dollar by 2017[3]. One of the direct consequences of such a success is the competition for wireless resources in order to provide the clients the best possible quality of service.

Both industry and academia have put a huge effort on how to improve wireless networking. New proposals for local and access networks are the main focus of several studies and, as a consequence, experimental as well as theoretical contributions are continuously arising. At the lower layers, for instance, new modulations and communication techniques have been developed to increase transmission rates and communication reliability[88]. All these efforts are pushed by the industry, which has identified the outstanding potential for business.

One of the approaches to achieve better performance in wireless networks is to rely on a *measurement-based strategy* to undoubtedly check system behavior and react accordingly[5;21;82]. Analyzing the wireless traffic uncovers problems hard to discover otherwise. To monitor the wireless traffic is also useful to study mobility[28;102;111], contacts in DTNs[84], and wireless network protocols[112].

5.1. Legacy monitoring methods

Several monitoring methods and tools that attempt to monitor wireless activity, as much faithfully as possible, have appeared in the latest years[10;24;26;41;55;66;91;105;107]. Initially, networks were mostly monitored through SNMP or logs recorded from the wired side of APs[10;41]. These solutions did not provide a complete view of the traffic on the medium: unassociated stations were basically ignored. Instrumenting all the nodes, possibly managed by different entities, in order to gather communication records is unfeasible. For evident coverage limi-

tations, these methods have been in past replaced by "passive" monitoring methods where some stations were dedicated to listen and record all the ongoing traffic.

Even for passive wireless networking measurements the completeness problem still remains. Monitoring nodes may lose events because of wireless physical issues, e.g., fading channels, interference, collisions, and hardware outages[24]. Therefore, relying on a single monitor is error prone and may lead to biased conclusions. As a consequence, typical monitoring systems employ multiple passive nodes scattered out across an area of interest to capture as much information as possible. The outcome of such a process is a collection of traces, each one from a different monitor, which are merged into a single file that provides a better and more complete picture of the wireless activity in the target area[100].

Using distributed monitors improves completeness but also introduces more complexity and scalability issues to the system. A distributed monitoring system increases the amount of data collected but imposes the utilization of merging techniques to come out with a coherent merged trace. Leaving aside storage limitations, merging traces can be cumbersome, as it introduces complex synchronization issues. Prior to the merging procedure, predefined heuristics are employed to circumvent clock drifts among the different traces. The monitoring system must shift the timestamps in the traces so as to make them coherent. This drift may not be constant throughout the traces and must be recomputed using received probes as references. After adjusting the timestamps, traces can be finally merged. Furthermore, in such a procedure, each single packet must be identified without ambiguity to be considered only once in the merged trace[92].

5.1.1. Large deployments

Much effort has been devoted to large-scale monitoring systems and merging techniques. Cheng et al. claim that the dynamics of a wireless environment can be only rebuilt if all frames and delivery outcomes are captured[24]. In this direction, typical approaches rely on the deployment of as many distributed passive monitors as possible. They use a central entity in charge of generating merged traces; they are also concerned with the contrasting requirements of completeness of the captured traces and scalability of the monitoring systems. Yeo et al. have been among the first to deal with monitor placement and capture quality[116]. DIST is a large-scale general-purpose WLAN monitoring system (210 monitors, with double radio interface) at the Dartmouth College campus separate from the WLAN infrastructure[107]. VISUM, developed at UC-Santa Barbara, is an extensible Java-based system with a real time presentation feature of various captured data statistics[42]. WizNet is a WLAN performance monitoring system built on 2.4 GHz off-the-shelf ZigBee sensors[123]. ZigBee devices have the advantages of low price and low battery power consuming. Pazl is a promising low cost and automated mobile crowdsensing based Wi-Fi monitoring sys-

tem[80]. Wizned and Pazl are essentially used to study the wireless signal and the covering respectively.

Even though these systems are sophisticated, *their main goal is simply to merge as many traces as possible, rather than improving the merging efficiency by previously selecting the most relevant traces.*

5.1.2. Trace inference

Inferring missing information in captured traces is a smart way to integrate traces. Jigsaw (system with 96 monitors) and Wit propose some heuristics to fill traces with missing elements[24;66]. These heuristics are based on the IEEE 802.11 protocol behavior. For example, if a monitor captures a RTS and a DATA frame from a station, it is easy to conclude that the station received a CTS before the DATA even if it has been missed by the monitor. If an ASSOCIATION RESPONSE sent by an AP is captured, it means that the AP preemptively received an ASSOCIATION REQUEST. Like this, a series of finite state machines can be built and, reading the trace in inverse temporal way, the states which are skipped correspond to missed packets.

This solution is still limited and correlated to the initial quality of the capture, e.g., if neither an ACK nor a retransmission is captured after a DATA frame, we cannot conclude if the frame was lost or just the ACK was not captured. For this kind of ambiguity, Jigsaw relies on a double level of inference: link-layer and transport layer inference. For the last example, we could check if a TCP ACK is transmitted to prove that the data link-layer frame has been actually well delivered.

5.2. IEEE 802.11 background

For the sake of completeness and because our proposals rely in the operation of the IEEE 802.11 protocol, we briefly present its structure and operation. We focus on the aspects that are directly related to our work.

The IEEE 802.11 standard denotes a set of four media access control (MAC) mechanisms (DCF, PCF, HCF-EDCA, HCF-HCCA) and a set of six physical layer (PHY) specifications (FHSS, DSS, IR, OFDM, HR/DSSS, ERP)[101]. In this section, we recall only some aspects of the IEEE 802.11 standard that will be useful in this work. For a complete view we refer to the whole standardization.

Devices equipped with a wireless local area network (WLAN) card, called wireless stations (STA), are federated in Basic Service set (BSS) or Independent BSS (IBSS). In the first case, stations communicate through a common access point (AP), they are so supported by a network infrastructure. In the latter case, stations can directly communicate among them in ad hoc mode.

Medium access mechanism

One of the most interesting aspects in WLAN is how the problem of the access to the medium has been tackled. Despite wired networks, additional problems arise: (*i*) higher error probability, (*ii*) dynamic topology, (*iii*) STAs have a limited view of the network, (*iv*) hidden stations, (*v*) shared medium.

Since we will mainly deal with ad hoc networks in this thesis, we focus on the Distribution Coordination Function (DCF) MAC. Since Wi-Fi devices are half duplex, DCF relies on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). We highlight the word "avoidance" and not "detection" as in wired networks. Before transmitting, a station senses the medium for ongoing communications. If the medium is idle, the STA transmits its frame. If not, the station waits a Distributed Inter-Frame Spacing ($DIFS$) time and, after the ongoing transmission ends, it calculates another waiting time as $Random(CW) \times collisionSlot$. $Random(CW)$ gives a number between 1 and $n-1$ of contention windows (CW). The additional waiting time, called backoff, is due to the presence of other STAs waiting to transmit. Stations with a lower waiting time start to transmit as soon as the medium is idle, the others cut this time from their waiting time.

Acknowledgment for point to point communications

Since a station cannot detect collisions while transmitting, the standard provides an ACK frame for point to point communications. In this case, a STA receiving a frame gives back an ACK frame to the transmitter. The ACK is sent after a Short Inter-Frame Spacing ($SIFS$) time ($SIFS < DIFS$). If a transmitter STA does not receive an ACK, it proceeds with a retransmission. The stations must gain the access to the medium as before, but this time the CW is doubled. In according to the Binary Exponential Backoff algorithm, at every retransmission CW is doubled until a CW_{max}. The standard provides values for CW_{min}, CW_{max}, $SIFS$, $DIFS$, $collisionSlot$, and the maximum number of retransmission essays, after which the upper layer will be informed about the frame rejection.

RTS, CTS, and NAV

In point to point communications, the standard defines an optional mechanism to reserve the medium for a specific time. Before transmitting a STA sends a Request to Send (RTS) frame to the destination STA. If this last answers with a Clear to Send (CTS) frame, then the sender transmits its frame and waits for an ACK (Figure 5.1). A frame exchange becomes a tuple (RTS-CTS-DATA-ACK). If this chain is broken, the whole procedure must restart. This mechanism also allows protecting frame exchange from hidden terminal collisions.

RTS and CTS frames have a duration field that defines the total exchange elapsing time (in μ seconds). Neighboring STAs thus know the duration of the following transmissions.

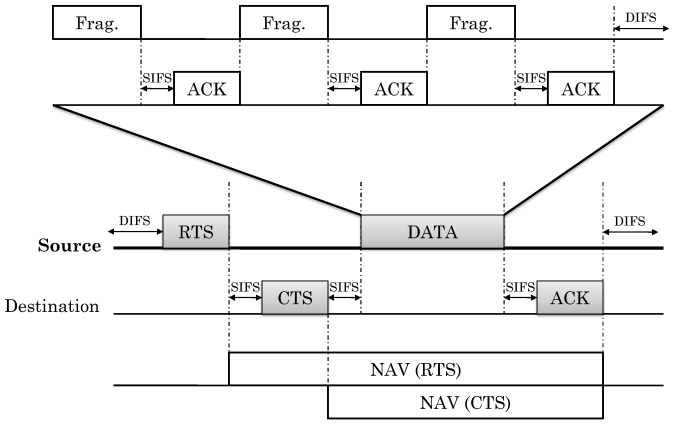


Figure 5.1: RTS-CTS mechanism and fragmentation.

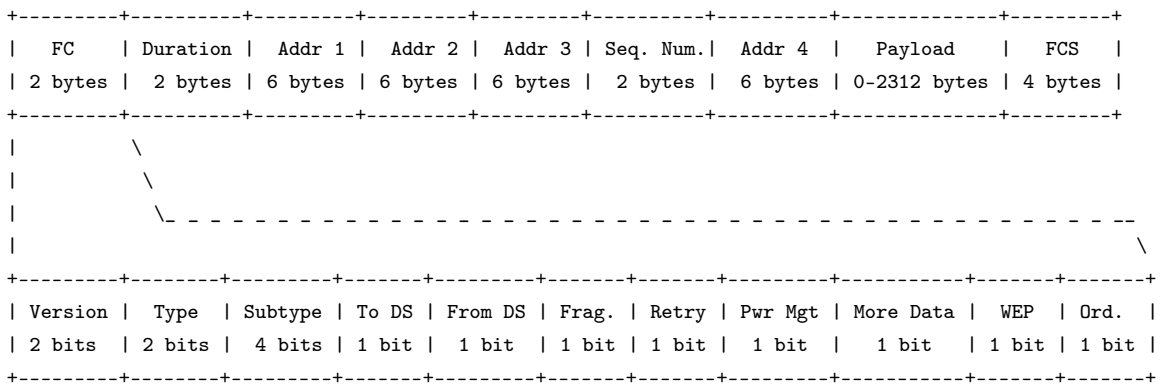


Figure 5.2: MAC frame format and frame control field structure.

They save these values in a Network Allocation Vector (NAV) and during these periods they cannot send any frame, neither a CTS. With NAV, STAs know, far in the future, if the medium has been reserved by RTS-CTS.

Fragmentation

In order to increase the transmission reliability, large frames are split into smaller frame fragments. Once a source gets the access to the medium, it is authorized to send fragments in a burst until the end or until an ACK relative to a fragment is missed. During a normal operation, a fragment is followed by a SIFS and by the corresponding ACK, as shown in Figure 5.1.

MAC frame format

As illustrated in Figure 5.2, the first two fields of a frame are two bytes for frame control and two bytes for the duration specification. The frame control fields, in turn, is composed

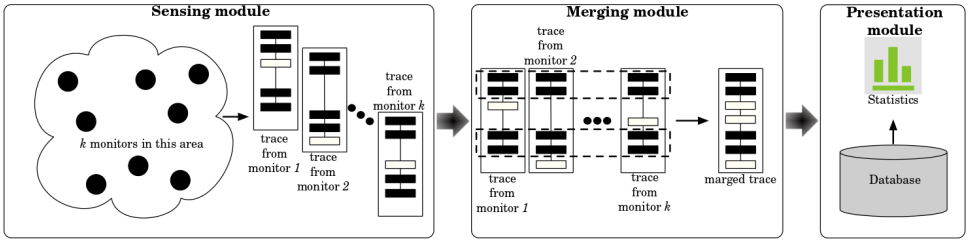


Figure 5.3: Typical wireless measurement system architecture: sensing, merge, and presentation modules.

by 11 subfields. The first three fields identify the version and the type of frame (control, management, data). Next two bits reveal the type of communication (infrastructure or ad hoc). Next two bits signal if this frame is part of a bigger fragmented frame and if it has been retransmitted. The sequential number field contains a 2 bytes value increased at each frame exchange. We use this counter to estimate the accuracy of a wireless traffic capture in Section 7.3.

5.3. Detailed problem statement

A wireless sensing system is generally composed of three main modules: sensing, merging, and presentation as shown in Figure 5.3. In the *sensing module*, monitoring nodes are responsible to collect frames it observes in the wireless medium.[1] As there are potentially several monitors, the output is a collection of traces obtained by different monitors. For sake of simplicity we assume that each monitor produces only one trace. The *merging module* is in charge of building the compiled trace using as input all the traces collected by the sensing module. Before merging, reference frames must be identified from unique frames. These unique frames embed a 64-bit timestamp and are transmitted only once, such as beacons and non-retransmitted probe responses. Finally, the *presentation module* is responsible to store previous measurements and to deliver statistics concerning the wireless network activity.

The main problems arising with this approach are the following:

- Sensing module scalability.
- Monitor placement.
- Merging module computational complexity.
- Biased measures due to corrupted traces.

[1] We consider wireless activity at the MAC layer. For this reason, a trace is a set of MAC traces that a monitor in promiscuous mode can "hear".

We will give particular attention to the first two problems, and they will be the focus of the following two chapters of this thesis. For the the sake of completeness, we briefly discuss below the other two problems.

5.3.1. Merging module computational complexity

The problem is that the merging module concentrates the main computational efforts of the system. Let $\mathcal{T} = \{t_1, ..., t_k\}$ be the set of k traces captured by k monitors during the same time interval, where each trace t_i is composed of v events (i.e., frames),[2] we can estimate that the merging procedure requires $O(k \times v)$. This can be more easily observed if we subdivide the merging procedure into at least the following tasks: reading traces, extracting unique frames, discovering reference frames, synchronizing traces, and writing the result into another trace as the final merge. These tasks are executed $k - 1$ times until the final single merged trace is achieved because the merging procedure is executed over pairs of traces ($t_i, t_j \in \mathcal{T}$) in a recursive fashion. Hence, the input of each merging procedure is the result of the previous one with another trace from $\mathcal{T}$.

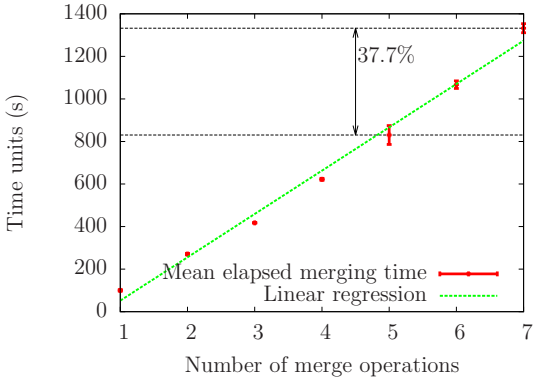


Figure 5.4: Time elapsed after a complete merging procedure for an increasing number of traces. Average time and 95% confidence interval are calculated over 10 experiments. Reducing the number of traces can significantly affect the time required for the entire operation.

The first task requires $O(v)$, as the identification of unique frames is performed in a single parse of the trace. These frames are inserted into a hash table, using their embedded timestamp as key. Therefore, whenever a hash collision occurs, a reference frame is discovered since the same frame is identified in both traces (this operation involves only two traces at time). The output lists the set of reference frames used for timestamp synchronization. To accomplish that, one of the traces has the timestamps of its reference frames adjusted according to the timestamps of the other trace. The remaining frames of the updating trace, on the other hand, are adjusted using a linear regression method. The synchronization task takes then $O(v)$ and the writing task into the output trace takes $O(v)$. Finally, the whole procedure requires $O(k \times v)$.

Although the merging procedure is linear, the number of events within traces can be huge, leading to a potentially very large total time required before convergence. Therefore, even the linear procedure may not be fast enough for merging traces. Our proposal presented in Chapter 6 reduces the amount of time needed for merging, independent of the

[2]We refer to the contents of a trace as "events" to keep the nomenclature as generic as possible.

algorithms employed, by reducing the number of merging operations. The main challenge, however, is the definition of the most appropriate metric and, furthermore, the criteria used to select the subset of the most representative traces.

To reinforce our arguments, we consider an experiment where we capture wireless traffic with eight monitors and traces are truncated at only 300,000 frames each. Figure 5.4 illustrates the time elapsed after a complete merging procedure. Although the problem is linear, the slope of the curve is greater than one. Thus, if we reduce the number of traces by only two (25%), we can save approximately 38% of the time. If we consider offline applications requiring timely responses from measurements, reducing the amount of time needed for merging is fundamental.

5.3.2. Biased measures due to corrupted traces

Trace corruption or adulteration could lead to biased measurements. The problem becomes trickier when there is no central measurement unit. This is exactly what happens with *wireless community networks*[8;86;117], where users, in exchange of something (e.g., connectivity), contribute with their own resources. The undeniable problem from users' participation is the possibility of malicious actions.

Counting on users' participation is a low cost alternative to improve the efficiency of a wireless measurement system. Nevertheless, users must receive incentives to participate[56]. Such incentives could be the possibility of improving the quality of their own wireless access network based on their behavior. The main counterpart of such a system is that *malicious users can see this as an opportunity to benefit themselves in detriment of the others or as an opportunity to simply disrupt the network operation*.

Users can feel motivated to insert fake traces into the monitoring system with the aim to achieve some rewards (i.e., virtual credits, micro payments, gadgets, improved connection quality, better reputation in the internal social system, etc.). This attack can have, as a consequence, either the attraction of additional infrastructure towards the malicious node or the purge of near infrastructure to other areas. We call these attacks, respectively, *attractive* and *repulsive* attacks (see Section 7.1 for a more formal definition of these types of attacks).

5.4. Summary

In this chapter, we browsed legacy wireless traffic monitoring architecture. They can be divided in active (with generation of polling and managing traffic) and passive (just listen and record everything passing through the medium). Passive methods can be also divided in large monitoring deployment, where a sizable fleet of monitors is installed in the area of interest, and trace inference, where missed frames in the traces are reconstructed following the IEEE 802.11 protocol behavior.

These approaches tend to save and process as much data as possible rather than capture and process the right data, with evident problems of scalability.

The main problems of traditional wireless monitor systems that we detect and we tackle in following of this work are:

1. **Scalability.** Quality of measurement is correlated to the quantity of monitors employed [99;116]. Due to cost issues, it becomes hard to get a wide area capture.

2. **Geographic distribution.** Monitors' range should overlap for redundancy purposes and, at the same time, they should cover the whole capturing area.

3. **Sensitivity.** Corrupted or adulterated traces may lead to biased measurements.

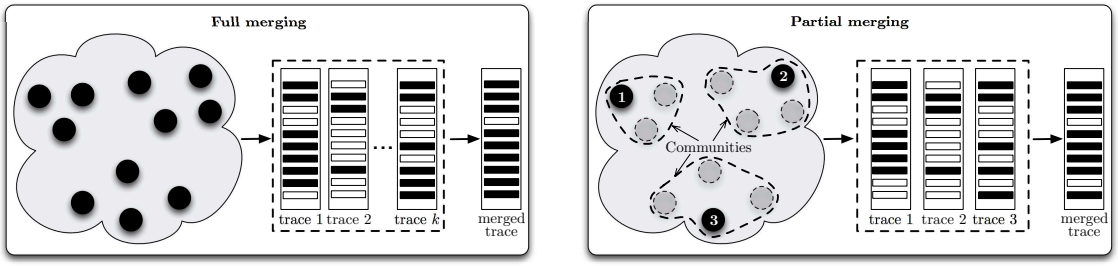
Chapter 6

Scalable wireless traffic capture

The first contribution in the area of WLAN monitoring is to design, deploy, and test new wireless traffic capturing methods. We try to maximize the amount of captured traffic keeping costs low. Costs are the sum of fixed costs and variable costs. Fixed costs increase when a new monitor must be purchased, installed, deployed, and maintained. We can consider that this cost is amortized if the new monitor significantly improves the capture process. A new monitor also means a new trace to process. Gathering, analyzing and merging together more data increases variable costs. Thus, also in this case we can consider that this cost is more or less amortized if the additional trace contains or not original useful data.

In relation to the approaches presented in Section 5.1, we address the following question: is it worth merging all the traces or a subset of them is enough? In fact, different traces likely bring their own specific observations. Nevertheless, traces might have a high level of similarity. As a consequence, it becomes possible to improve system scalability by reducing the number of traces to be merged while keeping the quality of the final fully merged trace.

The main idea behind our trace selection methodology relies on the notion of similarity between traces, which can be used as input of community detection algorithms. These algorithms find subsets of traces with high similarity (i.e., "communities of traces"). If we consider that at least one trace per community must be used in the merging procedure, we have a clue to the minimum number of traces to be used. This cutoff value is computed based on an additional criterion defined from experimental observations. The problem becomes then finding the exact traces to be used from each community. To accomplish that, we rank all the traces according to their individual contribution, to the final merged trace considering pairwise similarity values[96]. This ranking procedure stops when every trace is sufficiently different from the others, which happens when the number of ranked traces and communities match. Our results show that the proposed approach is efficient as the k top-ranked traces are from k different communities. The main idea is illustrated in Figure 6.1. In Figure 6.1(a), we depict the fact that the final merged trace is richer if we consider all the traces collected. The shortcoming in this case is the number of pairwise merging operations executed, which can be very large. On the other hand, Figure 6.1(b) shows that we can sig-



(a) All collected traces are merged. The final output has $\frac{9}{10}$ of all frames transmitted and is obtained after $k-1$ merging operations.

(b) Only the traces from the most representative monitors of each community are merged. The final output has $\frac{8}{10}$ of all the frames transmitted, but can be obtained after only 2 merging operations.

Figure 6.1: Impact of merging a selected subset of traces. In each trace, the collected and missed frames are represented by black and white rectangles, respectively. Merging all traces together permits more complete results but also incurs in more merging operations.

nificantly reduce the number of merging operations without losing too much information (wireless frames) if we properly select a representative subset of traces.

In a nutshell, our contributions can be summarized as follows:

- **Trace similarity analysis.** We propose five metrics to analyze the similarity between pairs of traces. Each trace is composed of a sequence of IEEE 802.11 frames organized in chronological order. These frames come from different communications (flows) within the same area. The level of similarity depends on the amount of flows or frames captured by both traces. Hence, the higher the intersection between the traces, the higher their similarity. The identified intersection can have different impacts on the metric depending on additional parameters, such as the rarity or the duration of the flow. We also evaluate these alternatives assigning weights to them.

- **Community identification.** We observe that although monitors are scattered out in the area of interest, their contributions to the final merged trace may overlap. We propose the utilization of community-based algorithms to find such "communities of traces", giving us a clue to the number of top-ranked traces to be used in the merging procedure.

- **Trace ranking.** We propose a method to rank individual traces according to their potential contribution to the merged trace. This ranking method models the problem as a fully-connected graph, where the vertices represent the traces and the edges between them are weighted according to pairwise trace similarity. Based on this model, we can rank the more relevant traces computing possible paths in the graph.

- **Monitor positioning.** A positive side-effect of our ranking method is the additional possibility to identify monitors not well positioned in the monitoring area. If we follow

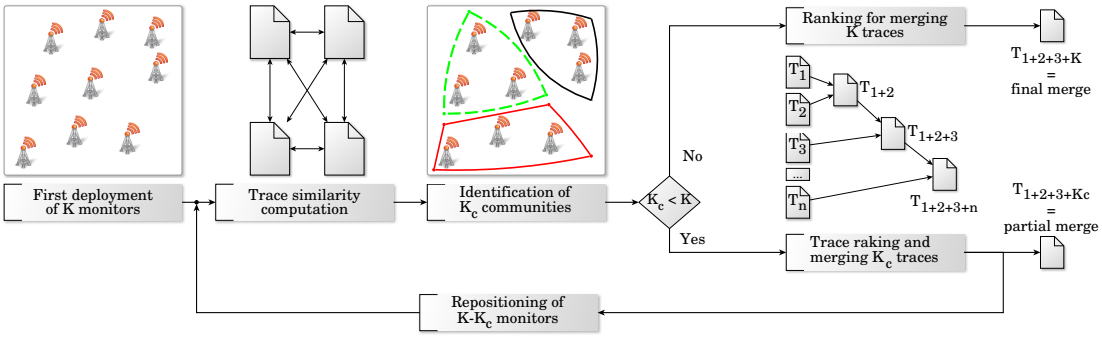


Figure 6.2: Additional tasks of the proposed merging procedure.

the list of ranked traces from bottom-up, we can have an idea of monitors that are badly placed. Therefore, we can have the list of appropriate candidates to be moved elsewhere.

- **Scalability gains.** We show results attesting that our method leads to scalability improvements, meaning that we can achieve a higher level of accuracy with a lower number of traces. Selecting a subset of traces can improve scalability as the procedure of finding the subset of the most representative traces help reduce the number of merging operations (which are costly). Although the subset selection procedure adds complexity to the system, it is executed less often then the whole merging procedure.

6.1. Improving Trace Selection

In our approach, we introduce additional tasks within the merging module. Three new tasks are added to reduce the number of traces to be merged. Without loss of generality, we assume that each monitor produces one trace. The proposed tasks, shown in Figure 6.2, are executed in the following sequence:

Step 1: similarity metric computation. This task receives the set of traces $\mathcal{T}$ and computes the pairwise similarity matrix between them. We test five possible metrics (explained in details in Section 6.2.3): *intra-* and *inter-flow*, *Adamic*, *Power*, and *Weighted inter-flow*. The first two metrics compute, respectively, the fraction of frames captured by both traces over the total number of frames and the fraction of flows captured by both traces over the total number of flows. The last three metrics provide different weights according to the number of common flows, the rarity of flows, and the number of frames per flow. The outcome of this task is modeled as a fully connected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where the set of vertices $\mathcal{V}$ represents $\mathcal{T}$ and edges in $\mathcal{E}$ have a weight proportional to the pairwise similarity between traces.

Step 2: community identification. Our graph model yields the observation of subsets of traces containing higher similarity. The identification of such subsets leads to the organization of traces in "communities". In practical terms, these communities are formed among

monitors producing similar traces. This is possibly a consequence of monitors' physical position since the closer the monitors, the more similar their traces. Upon running community detection algorithms, we have the number of communities, which is used as a cutoff value on the number of traces for merging. Such cutoff value is denoted by k_c, where $k_c \leq k$. In this work, we find communities by using three different algorithms: Walktrap, Infomap, and Label Propagation. These algorithms consider as a community the set of vertices connected through high number of edges. In our case, however, since the graph is fully connected, they consider as a community the set of traces connected through edges with higher weights.

Step 3: trace ranking. The ranking task receives as input the set of traces $\mathcal{T}$ and the number of communities k_c. Based on these parameters, the obtained traces are ranked in an ascending order of similarity as computed by a sorting algorithm. In our case, we compute the Travelling Salesman Problem solution path to obtain the sequence of traces. Since the weights of the edges connecting traces within different communities are lower, the minimum path becomes composed of consecutive traces from different communities. If $k_c = k$, we consider that all the monitors have captured representative traces and that they are all well located. The final merge, in this case, is composed of all the k traces. Unlikely, if $k_c < k$, the first k_c traces in the TSP solution sequence are used to set up the most efficient merge, since they are the most dissimilar traces. We denote the set of ranked traces as a sequence $\mathcal{S} = \langle t'_1, \ldots, t'_{k_c} \rangle$, where $t'_i \in \mathcal{T}$ and $t'_1 \preceq \cdots \preceq t'_{k_c}$. The remaining traces not included in $\mathcal{S}$ are from monitors that could be moved elsewhere to also contribute with representative traces. *Merging the traces in $\mathcal{S}$, we can reach an efficient balance between completeness and scalability of the sensing system.*

Algorithm 3 illustrates the sequence of tasks proposed to reduce the number of traces before merging. Note that the sequence of traces found is used as an input of the next task within the merging module.

Algorithm 3 Trace selection algorithm.

Require: $\mathcal{T} = \{t_0, \ldots, t_k\}$
Ensure: $\mathcal{S} = \langle t'_1, \ldots, t'_{k_c} \rangle$
  $\mathcal{G}(\mathcal{T}, \mathcal{E}) \Leftarrow$ `compute_metric_similarity`$(\mathcal{T})$
  $k_c \Leftarrow$ `identify_communities`$(\mathcal{G}(\mathcal{T}, \mathcal{E}))$
  if $k_c < k$ **then**
    $\mathcal{S} \Leftarrow$ `rank_traces`$(\mathcal{T}, k_c)$
  else
    $\mathcal{S} \Leftarrow$ `rank_traces`$(\mathcal{T})$
  end if

Discussion

The proposed trace selection algorithm can improve the scalability of sensing systems by reducing the number of traces to merge. The additional tasks add some complexity, but, once the ranking process shown in Figure 6.2 is concluded, it will only remain to start the usual sensing, merging and presentation modules, for all the next captures.

The similarity computation complexity, between two traces, may be considered as linear if equal events are detected with hash collisions, hashing traces events. The slower community detection algorithm, Walktrap, has a worst case time complexity of $O(e \times k^2)$ (where k and e are the number of vertices and edges in the input graph respectively)[79]. The Traveling Salesman Problem is NP-Hard. Nevertheless, we work on a fully connected graph satisfying the triangle inequality. Under these conditions, heuristics with a $O(log(k))$ complexity are available[93]. Although we include problems with complexity higher than linear, we argue that the number of traces is much smaller than the number of events per trace ($k \ll v$). As a consequence, the time required for convergence tends to be smaller than if we considered all the collected traces. Besides the input size, we advocate that heuristics exist, which can also bring down the overall complexity. In addition, the proposed trace selection algorithm can be run only during the first monitor deployment to settle it down. After that, it can be only rerun if a new monitor is installed or if a monitor repositioning is required. This is an additional heuristic to reduce the number of times the whole solution is executed.

6.2. Experimental Setup

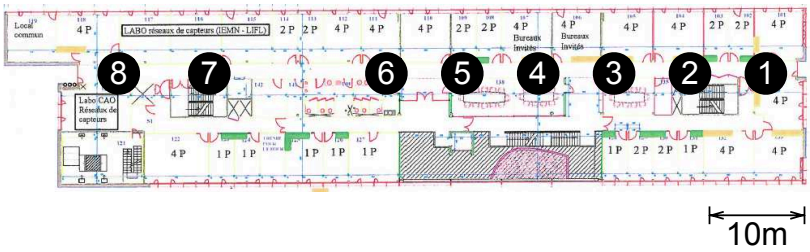
In this section, we describe the merging tool used in this work and the experimental scenarios.

6.2.1. Scenarios

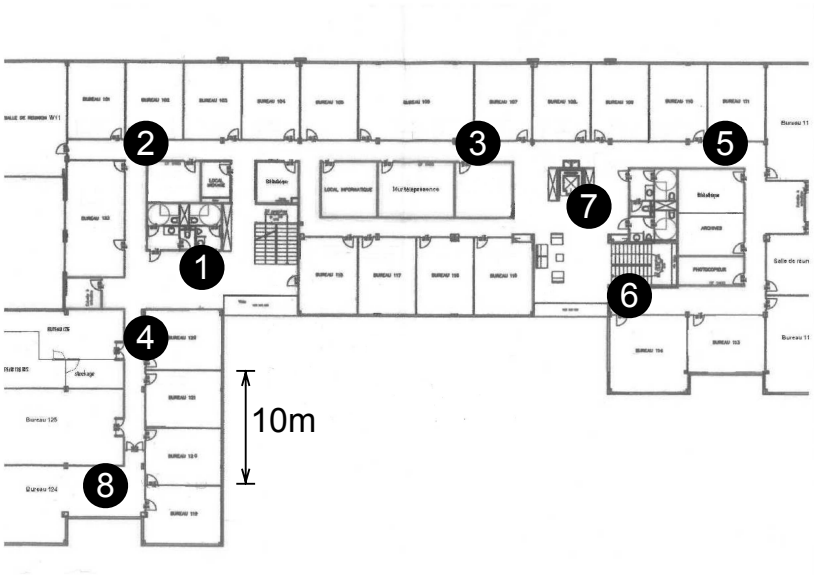
We have conducted experiments in two scenarios. We call the first one "IRCICA", as we deployed eight monitors at the second floor of the IRCICA/LIFL computer science laboratory of Lille[1]; and we call the second one "INRIA", as we conducted our experiment at the INRIA building also in Lille. Figure 6.3(a) shows the placement of monitors along the corridor at IRCICA (leading to a linear shape). Note that monitors 1 to 6 are equally spaced, while monitors 7 and 8 are slightly separated from the others. At INRIA, as shown in Figure 6.3(b), monitors are placed along the L-shaped floor. Monitors cannot directly view each other, as they are separated by walls.

In both scenarios, the nodes sensed the wireless network activity during 100 minutes, collecting IEEE 802.11b/g frames. The wireless traffic collected in both scenarios is com-

[1]Measures have been conducted in collaboration with the Inria FUN team in the context of the ANR Rescue Project.



(a) IRCICA scenario.



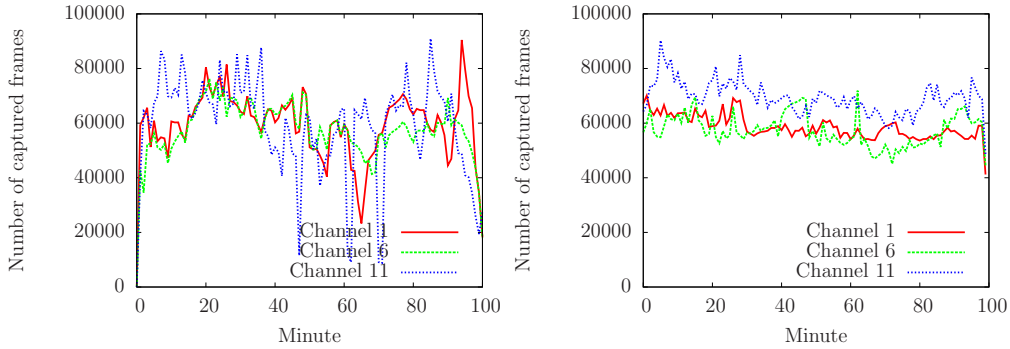
(b) INRIA scenario.

Figure 6.3: Monitor deployment for IRCICA and INRIA scenarios.

posed of control, management, and data frames to and from access points in the area. According to our measurements, we have collected frames to and from 37 and 9 access points in the IRCICA and INRIA scenarios, respectively. Figure 6.4 shows the amount of traffic captured in both scenarios after merging the traces collected by the eight monitors in the same time interval. We plot all the traffic captured on each IEEE 802.11b/g non-overlapping channels, i.e., channels 1, 6, and 11.

Note that the network activity is similar in all non-overlapping channels on each scenario. In this work, we decided to use channel 1 as our reference. Nevertheless, it is worth mentioning that the obtained results are not significantly affected by the operating channel. All trace sizes are reported in Table 6.1. In our experiments, the size of the captured frames is limited to 220 bytes and MAC addresses are anonymous.

Frame losses always occur independent from the sniffer hardware and software configuration[99]. In our experiments, we use Asus EEEPC-4G netbooks as sniffers. They are equipped with 512-MByte RAM and three USB Wi-Fi Netgear WG111v3 cards as wireless network adapters. The operating system is a Xandros OS with a customized kernel.



(a) IRCICA scenario. (b) INRIA scenario.

Figure 6.4: Wireless traffic characterization in IRCICA and INRIA scenarios.

6.2.2. Merging tool

The traditional approach to merging traces involves a previous synchronization, which finds identical frames according to their timestamps. After identifying such frames, they are inserted only once in the final trace. We use WiPal tool to do this job[25;26]. WiPal gets as input an arbitrary number of IEEE 802.11 PCAP traces from different sensing nodes and compute a merged PCAP trace as result. WiPal also uses additional software for trace synchronization and for extraction of reference frames. Given two traces T_i and T_j, WiPal effectuates the following steps to provide a merged trace in output:

Table 6.1: Trace size for IRCICA and scenarios.

Trace Size (Mbyte)		
Trace	IRCICA	INRIA
1	183	129
2	193	120
3	214	114
4	303	72
5	269	99
6	246	150
7	127	140
8	107	61

1. **Identifying reference frames:** A frame is said to be unique when it appears on the wireless medium once and only once for the whole measurement duration. WiPal considers every beacon frame and non-retransmitted probe response as unique frames due to the 64-bit timestamps they embed (these timestamps are not related to the actual timestamps used for synchronization). These frames are extracted from the input traces and then intersected. The intersection process first puts every unique frame of T_i in a hash table h, then does the same for T_j unique frames. If a collision occurs, a reference frame is found.

2. **Synchronization:** Synchronizing two traces means mapping trace one's timestamps to values compatible with trace two's. WiPal operates on windows of $w + 1$ reference frames and for each of

them (R_i) the process performs a linear regression using reference frames $R_{i-\lfloor w/2 \rfloor}, \ldots, R_{i+\lceil w/2 \rceil}$. Once the reference frames are synchronized, the two traces are synchronized too accordingly.

3. **Merging:** Frames from synchronized traces are copied to the output trace avoiding duplicates.

WiPal operates offline and has shown to outperform other tools available, which made us adopt this solution in our work.

6.2.3. Trace Similarity

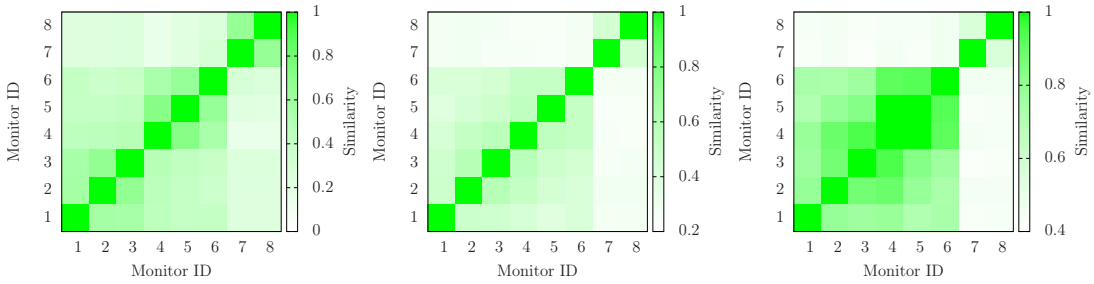
We propose five metrics to discover how similar two traces are. The *intra-flow similarity* computes the ratio of the number of frames simultaneously captured by two monitors over the total number of frames captured by them. The *inter-flow similarity*, on the other hand, considers the intersection of flows instead of frames. Therefore, it computes the ratio between the number of flows "observed" by the two monitors over the total number of flows "observed" by them. A flow is considered "observed" if at least one of its frames is captured. In addition, since flows may not be "observed" by all monitors and can have different numbers of frames, we give them different weights. The rarer the flow, the higher the weight assigned to it by the similarity metric. Likewise, the higher the number of frames in a flow, the more important it is for the similarity metric. This rationale is followed in both the *Adamic* and *Power* similarity metrics. They explore the rarity principle by assigning more weight to the similarity of traces sharing more rare flows. Finally, the *Weighted inter-flow similarity* assigns an additional weight to the number of frames in common per flow.

We consider that each trace is composed of flows of frames denoted by f. In addition, we consider that f_i^m is the mth source-destination flow in trace t_i and that p_i^n is the nth frame in t_i. Finally, we denote the cardinality of a set with $|\bullet|$.

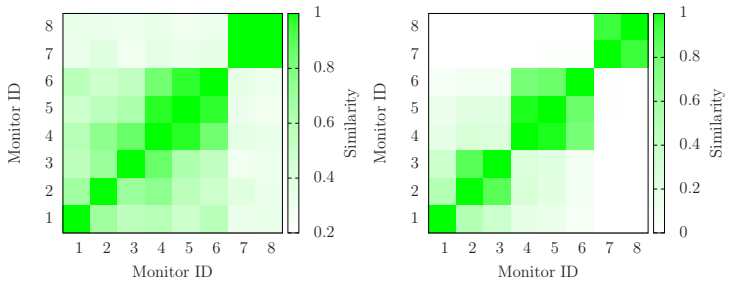
Intra-flow similarity

We use the Jaccard similarity index to compute the Intra-flow similarity. This metric considers all the frames captured by two monitors, digging into each source-destination flow. This is why we consider this metric as intra-flow. Considering two captured traces t_i and t_j as a set of frames, i.e., $t_i = \{p_i^0, \ldots, p_i^n\}$ and $t_j = \{p_j^0, \ldots, p_j^{n'}\}$, the intra-flow (Intra) similarity is computed as follows:

$$\text{Intra}(t_i, t_j) = \frac{|\{p_i^0, \ldots, p_i^n\} \cap \{p_j^0, \ldots, p_j^{n'}\}|}{|\{p_i^0, \ldots, p_i^n\} \cup \{p_j^0, \ldots, p_j^{n'}\}|}. \tag{6.1}$$



(a) Intra-flow similarity matrix. (b) Inter-flow similarity matrix. (c) Adamic similarity matrix.



(d) Power similarity matrix. (e) Weighted inter-flow similarity matrix.

Figure 6.5: IRCICA scenario, similarity matrices. (a) Intra-flow, (b) Inter-flow, (c) Adamic, (d) Power, (e) Weighted inter-flow.

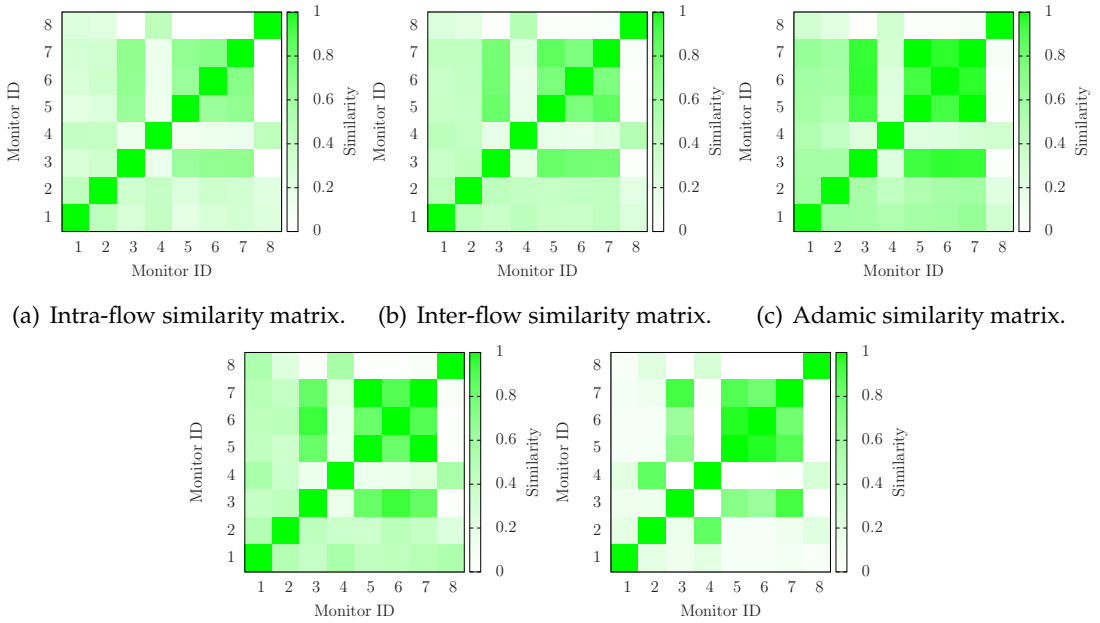
Note that $n \neq n'$ because the number of frames in t_i and t_j may be different.

Figure 6.5(a) depicts the intra-flow similarity matrix of the traces from IRCICA. Each point (i, j) is gradually colored according to its value of Intra(t_i, t_j). As the monitors have been sequentially placed, higher values are near the diagonal. In the figure, we can identify three geographical regions: a central region (monitors 4-5-6) and two side regions (monitors 1-2-3 and 7-8). As monitors 7-8 are slightly isolated from the others (see Figure 6.3(a)), they present a high similarity in opposition to the low similarity compared with all the others. This means that small changes in the geographic placement of monitors have a considerable impact on the amount of original data captured. In the INRIA scenario, shown in Figure 6.6(a), traces 1 and 2 share a low intra-flow similarity with all the others, whereas the remaining traces are split into two sets, 3-5-6-7 and 4-8.

Inter-flow similarity

We also use the Jaccard index to compute the Inter-flow similarity. In this case, we consider the traces t_i and t_j as a set of flows, i.e., $t_i = \{f_i^0, \ldots, f_i^n\}$ and $t_j = \{f_j^0, \ldots, f_j^{n'}\}$. Then, the inter-flow (Inter) similarity of traces t_i and t_j is computed as follows:

$$\text{Inter}(t_i, t_j) = \frac{|\{f_i^0, \ldots, f_i^n\} \cap \{f_j^0, \ldots, f_j^{n'}\}|}{|\{f_i^0, \ldots, f_i^n\} \cup \{f_j^0, \ldots, f_j^{n'}\}|}. \tag{6.2}$$



(a) Intra-flow similarity matrix. (b) Inter-flow similarity matrix. (c) Adamic similarity matrix.

(d) Power similarity matrix. (e) Weighted inter-flow similarity matrix.

Figure 6.6: INRIA scenario, similarity matrices. (a) Intra-flow, (b) Inter-flow, (c) Adamic, (d) Power, (e) Weighted inter-flow.

Figure 6.5(b), concerning the IRCICA scenario, shows that traces are divided in two blocks of 1-2-3-4-5-6 and 7-8, sharing a low similarity. In the INRIA scenario, on the other hand, there is no appreciable difference compared with the intra-flow similarity metric. This suggests a correlation between flows and frames captured in more distributed scenarios.

Adamic similarity

The Adamic metric was originally proposed to evaluate the similarity between plain texts[6]. We can consider traces also as a text file which lists flows of frames. Hence, we compute the similarity between traces t_i and t_j as follows:

$$\text{Adamic}(t_i, t_j) = \sum_{f \in \{t_i \cap t_j\}} \frac{1}{\log(\phi_f)}. \tag{6.3}$$

We denote ϕ_f as the number of times a given flow f appears in all traces. This number is called the frequency of f in $\mathcal{T}$. Thus, if a flow f belongs to the intersection between t_i and t_j, we take it into account for the Adamic metric computation. The first step is to count the number of times it appears in all traces. For instance, if $f \in \{t_1 \cap \cdots \cap t_k\}$, then $\phi_f = k$. Otherwise, if f appears only in those two traces, t_i and t_j, then $\phi_f = 2$. Note that considering the inverse of ϕ_f in the metric, the weight of a flow becomes proportional to its rarity in the collected traces. As a consequence, the similarity of the traces sharing rare flows is higher

since it is assumed that they are likely from a near location. As the range of the Adamic metric is not upper bounded, we normalize the values.

Figures 6.5(c) and 6.6(c) show the same sets of traces discovered with the *inter-flow similarity*. In this case, however, we note that they present a stronger inner connection as the color is darker. This means that traces with high similarity tend also to share rare flows.

Power similarity

Although the Adamic similarity takes into consideration the flows in common and their rarity, one may want to further increase the rarity impact. The Power similarity metric (Power) presents a power function in place of the logarithm function to compute the similarity of traces t_i and t_j. Hence,

$$\text{Power}(t_i, t_j) = \sum_{f \in \{t_i \cap t_j\}} \frac{1}{\phi_f^p}, \tag{6.4}$$

where we assume that $p = 3$, similarly to Cunche et al.[28]. We also normalize the metric.

In Figure 6.5(d), we show that in the IRCICA scenario, the metric identifies two sets of traces with high similarity, i.e., sets 7-8 and 4-5-6. In the INRIA scenario, on the other hand, the metric still produces the same result obtained by the previous metrics.

Weighted inter-flow similarity

From a higher point of view, it is worth considering the flows shared between two traces. The importance of each flow is proportional to the number of frames it has. Let us consider $\mathcal{T}$ the *corpus* of traces and each unique flow f_i^m as a single term in a trace t_i. Note that both f_i^m and t_i are sets of frames, where $f_i^m \subseteq t_i$. We weight the importance of a flow using the Term Frequency-Inverse Document Frequency (TDF) metric[85]. TDF is widely used in information retrieval and text mining fields. In our context, documents are traces and terms are flows. The Weighted inter-flow (W-Inter) similarity between two traces is computed as follows. A vector of TDF is assigned to each trace. The m elements of the vector are the products of two factors. The first one is the flow frequency in that trace, whereas the second is the logarithm of the inverse of the trace (the one containing that flow) frequency over all the traces in the corpus.

$$\text{TDF}(t_i, m) = \frac{|f_i^m|}{|t_i|} \cdot \log \frac{|\mathcal{T}|}{|\{t_i \subseteq \mathcal{T} | f_i^m \subseteq t_i\}|}. \tag{6.5}$$

The Weighted Inter-flow similarity (W-Inter) is a value in the range [0;1] (from orthogonal traces to equal ones), given by the cosine of the angle between these vectors. This is equal to the dot product of the vectors, divided by the product of their magnitude:

$$\text{W-inter}(t_i, t_j) = \frac{\sum_m \text{TDF}(t_i, m) \cdot \text{TDF}(t_j, m)}{\sqrt{\sum_m \text{TDF}(t_i, m)^2} \cdot \sqrt{\sum_m \text{TDF}(t_j, m)^2}}. \tag{6.6}$$

In Figures 6.5(e) and 6.6(e), we can observe how the inter-flow similarity metric can clarify the relationship between traces. In the IRCICA scenario, Figure 6.5(e), we can clearly distinguish a first cluster of high similarity traces from monitors 7-8, which correspond to the monitors located on the west side of the building. Because they are slightly separated from the other monitors, they present low similarity with them. Central monitors 4-5-6 compose another cluster, while on the east side of the building, we have a set of monitors 2-3 and a singleton with monitor 1. This last monitor produces a trace with a perceptible similarity with trace 2 fading down up to trace 6. We remark that, even if very geographically close, trace pairs 1-2 and 3-4 do not present a very high similarity among them.

Figure 6.6(e) shows that the traces from monitors 3-5-6-7, placed in the east side of the INRIA building (Figure 6.3(b)), have a very high inter-flow similarity as well as traces from monitors 2-4. Monitors 1 and 8 constitute two singletons.

6.3. Community detection

After computing the similarity between traces, we aim at detecting communities among them. In this section, we analyze the similarity metrics to find the one which better reveals community relationships among traces. For each metric in Section 6.2.3, we build the adjacency matrix of a graph, using the similarity values found earlier between each pair of traces as edge weight.

We use three different algorithms for community detection (or graph modularity discovery): Walktrap[79], Infomap[94], and Label Propagation[81].

The evaluation of the goodness of the graph division made by a community detection algorithm relies on the Newman and Girvan's *modularity* metric[74]. Assuming that the algorithm finds k communities, the modularity is defined as:

$$\text{modularity} = \sum_{i=1}^{k} (w_{ii} - a_i^2), \tag{6.7}$$

where w_{ij} is the element of a $k \times k$ matrix. w_{ij} is equal to the weight fraction of all the edges linking vertices from community i to community j, over all the weights in the graph, and $a_i = \sum_{j=1}^{k} w_{ij}$ is the weight fraction of all the edges touching nodes in community i. If the

Table 6.2: Walktrap modularity values and relative communities for all the similarity metrics.

Scenario		Intra-flow	Inter-flow	Adamic	Power	W-inter
IRCICA	Modularity	0.195	0.195	0.146	0.165	**0.239**
	Communities	No	No	No	No	[1,2,3] [4,5,6][7,8]
INRIA	Modularity	0.223	0.193	0.172	0.185	**0.337**
	Communities	[1,2,4,8] [3,5,6,7]	No	No	No	[1,2,4,8] [3,5,6,7]

Table 6.3: Infomap modularity values and relative communities for all the similarity metrics.

Scenario		Intra-flow	Inter-flow	Adamic	Power	W-inter
IRCICA	Modularity	-0.004	-0.0007	-0.019	-0.005	**0.197**
	Communities	No	No	No	No	[1,2,3,4,5,6] [7,8]
INRIA	Modularity	-0.005	-0.013	-0.0004	-0.016	**0.395**
	Communities	No	No	No	No	[1,2,4,8] [3,5,6,7]

Table 6.4: Label propagation modularity values and relative communities for all the similarity metrics.

Scenario		Intra-flow	Inter-flow	Adamic	Power	W-inter
IRCICA	Modularity	-0.004	-0.0007	-0.019	-0.005	**0.362**
	Communities	No	No	No	No	[1,2,3] [4,5,6] [7,8]
INRIA	Modularity	0.179	0.137	-0.0004	0.123	**0.261**
	Communities	[1,2] [3] [4] [5] [6] [7] [8]	[1,4] [2] [3] [5] [6] [7] [8]	No	[1,2,3,5,6,7] [4,8]	[1,2,4] [3] [5] [6] [7] [8]

community detection algorithm is not able to find any community structure or the partition can be assimilated as a random one, then $modularity \leq 0$; otherwise it is positive and upper bounded by 1 for graphs with a very clear and strong community structure.

6.3.1. Algorithms

Walktrap. It is a hierarchical agglomerative community detection algorithm. It starts considering every single node as a community and then it iteratively advances merging communities together. The idea behind the merging is that, once a distance metric between nodes is defined, short random walks tend to remain in the same community. The process ends when only one community, containing all the nodes, is achieved, but the best split is the one which maximize the modularity metric.

Infomap. As Walktrap, Infomap is based on random walks. At the beginning, each node has a unique Huffman code. The trajectory of a random walk, given by the sequence of visited nodes, will form a code of a certain length. The algorithm detects communities in order to minimize the code describing the flow of random walks.

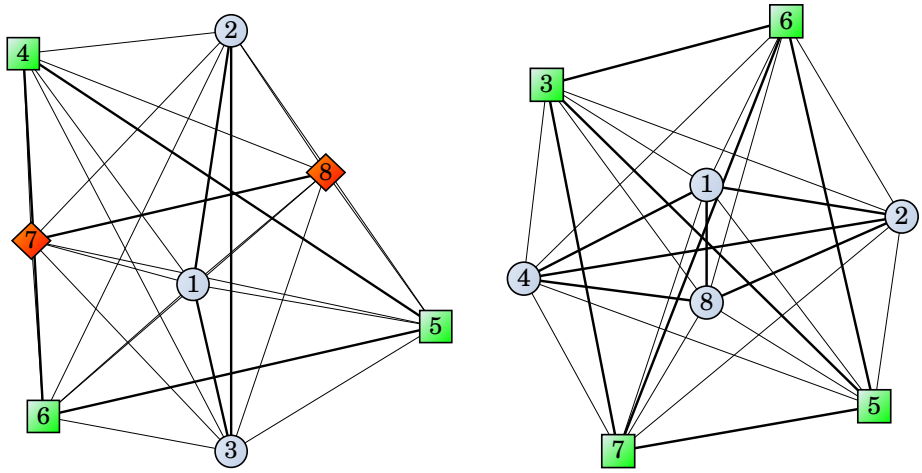
Label propagation. It is a hierarchical agglomerative algorithm too, but it relies on the concept of node neighborhood. At the beginning, all nodes have a unique label. During the iterative steps, nodes change labels, synchronously or asynchronously, according to the most popular label in their neighborhood. The algorithm stops when no further label changes are observed.

6.3.2. Results for community detection

We rely on the modularity metric to identify both the best similarity metric and the best community division of the graph. Table 6.2 shows the community detection results for the Walktrap algorithm. In the IRCICA scenario, all the modularity values are positive, but the highest is related to the weighted inter-flow similarity. With this similarity metric, the algorithm is also able to identify three communities while, with the other metrics, all the nodes are part of the same unique community. For the INRIA scenario, Walktrap finds the same two communities both with the intra-flow and the weighted inter-flow values. Nevertheless, in the latest case, the modularity metric is higher.

Results for Infomap are shown in Table 6.3. For both scenarios, only the use of the weighted inter-flow similarity metric presents positive modularity values. With all the other metrics, the algorithm is not able to find any community.

Label propagation presents only one positive modularity value in the IRCICA scenario (Table 6.4). It is also higher than the respective Walktrap or Infomap values. In the INRIA scenario, instead, all but the Adamic similarity metric has a positive modularity. The weighted inter-flow still presents the highest value.



(a) IRCICA scenario. (b) INRIA scenario.

Figure 6.7: Graphs generated using traces as nodes and weighted inter-flow similarity values as edge lengths. Different node shapes (◆, ●, ■) denote different communities.

The weighted inter-flow similarity metric is candidate to be the best similarity metric among the proposed ones as, based on this metric, all the community detection algorithms present the highest modularity value and often it is the only positive. With the same criteria, we consider the following as the best community division:

- **IRCICA:** [1,2,3]-[4,5,6]-[7,8]
- **INRIA:** [1,2,4,8]-[3,5,6,7]

6.4. Trace ranking

Merging traces from different monitors assumes that a monitor might capture an event that another monitor misses. Merging together many traces, however, is a CPU – and time-consuming process. Hence, depending on the sequence and the number of traces to merge, this procedure may converge faster when similar traces (i.e., traces with several equal frames) are not considered in the computation. To tackle this issue, we propose a method to improve the selection of traces to merge. As a final remark, we show that the procedure improves the system scalability without impacting on the monitoring information.

Strategy

For both scenarios, we consider the matrix of weighted inter-flow similarity values, calculated in Section 6.2.3 and shown in Figures 6.5(e) and 6.6(e), as a weighted adjacency matrix of a fully connected graph. In this graph, each vertex v_i corresponds to a captured trace t_i and each edge e_{ij} is linearly weighted proportionally to the similarity value between

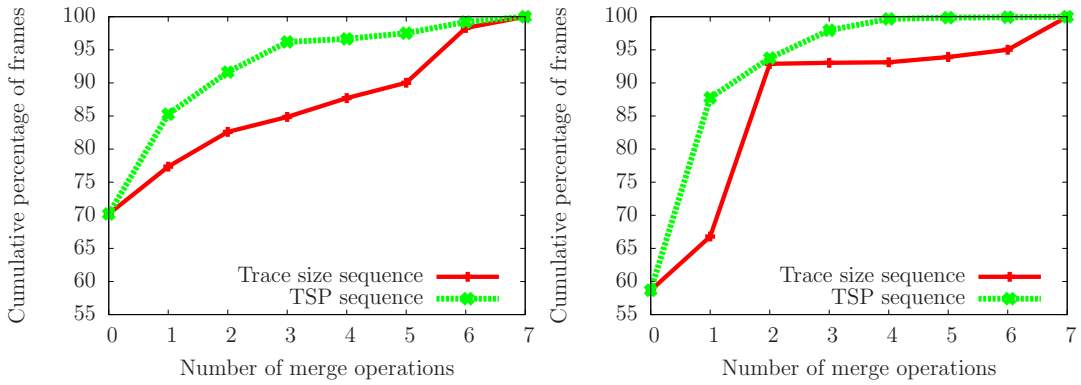
traces t_i and t_j (W-inter(t_i, t_j)). For the sake of visualization, we consider the length of each edge as proportional to its weight. The generated graph is a 3D graph. To respect all the distance relationships among nodes on a 2D representation, we use the ForceAtlas2 algorithm embedded in Gephi, a graph visualization and manipulation software, to lay out the graph[12].

Figures 6.7(a) and 6.7(b) show how nodes are arranged on a plane, in IRCICA and INRIA scenarios, respectively. In order to represent communities, nodes belonging to the same community have the same shape. We observe that pairs of vertices with high similarity (the ones belonging to the same community) are likely placed farther away, whereas vertices with lower similarity (vertices belonging to different communities) are placed closer.

In Figure 6.7(a), nodes of the community represented with a square $\blacksquare^{(4,5,6)}$ are placed very far from each other and nodes belonging to the diamond community $\blacklozenge^{(7,8)}$ are placed at the opposite respect the graph's barycenter. Nodes of the last community $\bullet^{(1,2,3)}$, are deployed along the vertical axis, at the middle of the graph as they must be far from each other, but close to all the other nodes at the same time. Node 1 falls close to the barycenter because, although it has been grouped with nodes 1 and 2, it does not share an high similarity with them and even less with the other ones (Figure 6.5(e)). Thus, considering a node, in its immediate proximity there are nodes belonging to different communities. The same trade off explains the dichotomy shown by the graph in Figure 6.7(b) related to the INRIA scenario. Nodes of the community $\blacksquare^{(3,5,6,7)}$ are placed very far from each other and at the opposite respect to the barycenter. Nodes of the second community $\bullet^{(1,2,4,8)}$ form a rhombus shape in the middle as they must be far from each other but close to the first ones. Nodes 1 and 8 fall in the middle as they are part of the second community but they do not have a very high similarity (Figure 6.6(e)).

Our ranking strategy is the following: considering the weighted adjacency matrices with the similarity values, we solve the Traveling Salesman Problem with the Concorde TSP Solver[27]. The solution represents the *order of traces to merge*. This is because it ranks traces according to their contribution to the final merge. The TSP solution path, in fact, sequentially touches closer nodes, which share a low similarity, and which give more new unique frames to the merged trace. This is especially true until a merge step equal to the number of communities found.

At the end, we can select a subset of traces with higher contribution to be merged and we can also identify the monitors not satisfactorily contributing to the measure. Merging a subset of traces can improve the system scalability while moving nodes to other positions can improve the monitoring system performance by extending the monitored area.



(a) IRCICA scenario. (b) INRIA scenario.

Figure 6.8: Comparison of the merging process performance ranking traces according to the size and the TSP solution.

6.5. Evaluation

6.5.1. Proposed strategy vs. trace size

We evaluate our ranking strategy comparing its performance with the sequence of traces sorted in reverse size order. This last ranking method could be reasonable to use in absence of different methods proposed in literature, as the goal is to have the most complete merged trace with less traces.

The trace size reverse sorting for the IRCICA scenario is:

$$\underbrace{4-5-6}_{\blacksquare}-\overbrace{3-2-1}^{\bullet}-\underbrace{7-8}_{\blacklozenge}.$$

The first three nodes are exactly the nodes of the community denoted by squared nodes in Figure 6.7(a). The following three nodes also compose the second community (circle nodes) and the last two (diamond) nodes constitute the last community. The TSP solution, from the same starting node is:

$$\underbrace{4}_{\blacksquare}-\overbrace{7}^{\blacklozenge}-\underbrace{1}_{\bullet}-\overbrace{6}^{\blacksquare}-\underbrace{3}_{\bullet}-\overbrace{5}^{\blacksquare}-\underbrace{8}_{\blacklozenge}-\overbrace{2}^{\bullet}.$$

The first three nodes belong to the three different communities. Next nodes are taken jumping from community to community. Figure 6.8(a) shows the performance of the above-mentioned methods. The TSP sequence achieves in three steps (four traces) more than 95% of the total traffic against six steps (seven traces).

For the INRIA scenario, the trace size reverse sorting is the following:

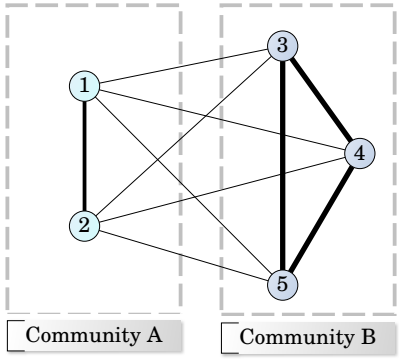


Figure 6.9: A fully connected graph with 5 nodes belonging to 2 communities. Higher the similarity between traces, thicker the edges connecting them.

$$\underbrace{6-7}_{\blacksquare}-\overbrace{1-2}^{\bullet}-\underbrace{3-5}_{\blacksquare}-\overbrace{4-8}^{\bullet},$$

while the TSP sequence from the same starting node is:

$$\underbrace{6}_{\blacksquare}-\overbrace{1-8}^{\bullet}-\underbrace{5-7}_{\blacksquare}-\overbrace{4}^{\bullet}-\underbrace{3}_{\blacksquare}-\overbrace{2}^{\bullet}.$$

The most relevant difference is the choice of the first two nodes. Since two communities have been detected, TSP selects one node per part. This choice leads to a 20% difference between the two strategies at the first merge operation, as shown in Figure 6.8(b). Trace 8 is the third trace chosen by TSP and it gives almost 7% of contribution over the total captured traffic. It gives the same important contribution in the reverse sorting strategy, but it is chosen only at the last merging operation, while the four previous traces do not give any important contribution to the merge. Moreover, the TSP selection covers all the wireless traffic, only merging the first five traces. This allows relocating the last three monitors, further enlarging the monitored area.

6.5.2. Proposed strategy vs. node degree

We consider as node's degree the sum of the weights of the edges exiting from that node, $\delta(v_i) = \sum_{j \in V} e_{ij}$. Each node has a degree value composed by two main contributions (Figure 6.9). The first contribution is given by all the edges' weights connecting the given node with others in the same community. These values are high and similar. Also edges' weight values in other communities will be high and similar among them, but sufficiently different from other communities, otherwise they would be part of the same community. The second contribution is given by the edges' weights connecting the given node with

nodes of different communities. As these values are much lower than the first ones, sorting nodes in according to the degree, will give a sequence composed by all the nodes of one community first, then all the nodes from a second community and so forth.

Using the weighted inter-flow similarity values, the ascending degree order for both scenarios is:

- **IRCICA:**

$$\underbrace{7-8}_{\blacklozenge}-\overbrace{\overbrace{1-2-3}^{\bullet}}-\underbrace{6-4-5}_{\blacksquare}$$

- **INRIA:**

$$\underbrace{1-8-2-4}_{\bullet}-\overbrace{3-5-6-7}^{\blacksquare}.$$

TSP solutions, from the same starting nodes, are:

- **IRCICA:**

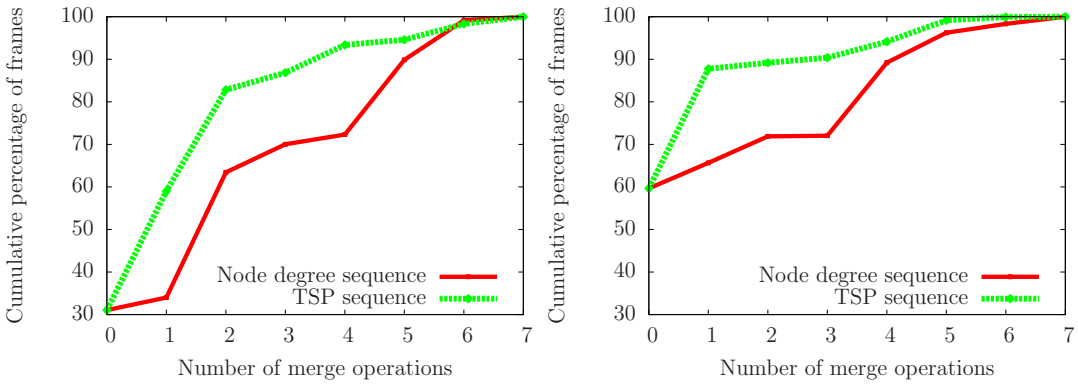
$$\underbrace{7}_{\blacklozenge}-\overbrace{6}^{\blacksquare}-\underbrace{3-1}_{\bullet}-\overbrace{5}^{\blacksquare}-\underbrace{8}_{\blacklozenge}-\overbrace{2}^{\bullet}-\underbrace{4}_{\blacksquare}$$

- **INRIA:**

$$\underbrace{1}_{\bullet}-\overbrace{6-3}^{\blacksquare}-\underbrace{4}_{\bullet}-\overbrace{7}^{\blacksquare}-\underbrace{8}_{\bullet}-\overbrace{5}^{\blacksquare}-\underbrace{2}_{\bullet}.$$

Figure 6.10(a) shows, for the IRCICA scenario, the comparison between the merging processes according to the ascending degree sequence and the TSP solution starting from the same node. With the sequence based on the degree, the first merge operation does not generate a relevant improvement because traces 7 and 8 compose a unique community. The second merge operation regards merging trace 1 that is part of a new community. The increment is around 30%. Next two operations finish that community with a total increment of around 6%. Changing again community, trace 6 gives an increment of about 20% at the fifth merging step. The greatest increments appear when the new traces to merge are taken from a new community. The TSP sequence naturally jumps from community to community, achieving about 82% in only two steps (corresponding to three traces, just as the number of communities).

Figure 6.10(b) exhibits the same behavior for the INRIA scenario. The two communities, discovered by the community detection algorithms in Tables 6.2 and 6.3, can be identified



(a) IRCICA scenario.

(b) INRIA scenario.

Figure 6.10: Comparison of the merging process performances ranking traces according to the ascending node degree and the TSP solution.

looking to figure 6.10(b) too. The ascending degree order presents only a remarkable increment at step four where a trace from a new community is merged, while the trace merged at step three does not produce any increment as it was the last one merged from the first community. The TSP sequence also presents a unique remarkable increment, but earlier, at step one, achieving almost 90% of the captured traffic.

6.6. Summary

It is difficult to define a rule to establish the exact number and position of Wi-Fi monitors to capture traffic in a target area. Their capturing range should overlap for redundancy purposes, so a high number of monitors is needed to cover a wide area. In this chapter, we have showed how, starting from an initial deployment, we can select monitors whose traces are actually important to merge together.

We propose five metrics to score the similarity among traces. We create a graph having traces as nodes and edges with a length proportional to the similarity value of the nodes they connect. On such a graph we discover the presence of clusters with community detection algorithms. These algorithms tell us how many traces are really important (as much as the number of communities detected) and which is the best similarity metric (the one producing the highest value of modularity after the division in communities). As a result, we know how many traces we need to merge and how many nodes we can move somewhere else. The subset of traces to be merged can still provide high traffic coverage, whereas the remaining traces belong to monitors that can be relocated to enlarge the target area. In order to select which traces to merge and which to move, we solve the Traveling Salesman Problem on the graph. Its solution, a sequence of nodes, is the rank of traces. This process can be iterated until the number of nodes is equal to the number of communities or, in the worst case, until

all the number of nodes is equal to the number of monitors. This last possibility means that all the traces must be merged to have a good capture quality.

Chapter 7

Sensitivity to input traces

In a collaborative measurement system, where network users themselves contribute to the monitoring activity, the ratio between the number of stations and monitors should be to the designer's discretion depending on his needs, giving space for low-cost and scalable solutions. The problem of monitor placement is avoided too as we expect to have enough density in most places due to the proliferation of wireless devices. Also, since the only requirement is to run a packet sniffer for his own benefit, we assume this should not be cumbersome for most users. It is worth mentioning that an active packet sniffer can run in background, consuming less than 1.5 kB of virtual memory, which includes the code, data, shared libraries, and used memory pages.[1] Users' willingness can also be compensated with incentives, such as resource allocation, micro payments, and higher reputation based on the contribution level.

We propose a collaborative method to improve the accuracy of a wireless network measurement system both considering time and space. In the proposed system, the achieved trace accuracy is correlated to spatial information to give a hint about overloading conditions in a certain area. Based on such information, it is possible to better adapt the network by moving additional infrastructure to overloaded areas. In order to improve the trace accuracy, wireless users collaborate to the system. Even though there are many signal processing techniques as well as many upper layer mechanisms for error detection and recovery, data losses are likely to occur especially when dealing with large and dynamic scenarios. Therefore, merging an increasing number of individual traces collected by distributed sensors can substantially improve the accuracy of the final trace. A user can indirectly improve the quality of its access network by collaborating to the wireless self-organizing network.

We conduct experimental tests using two different scenarios, called *collocated* and *scattered*, with different density of sensing nodes. In both scenarios, we show that every individual trace improves the accuracy (defined in Sec. 7.1.1) of the merged one and that the geographical distribution of sensing nodes can also impact on the final result. In addition,

[1] Test conducted using `tcpdump` on a Debian machine.

by introducing fake traces, we show that the detecting system can recognize a malicious node. In summary, the main contributions of this approach are three-fold:

- We propose a collaborative system to improve the accuracy of wireless measurement systems: We show that the accuracy of a given trace increases with the number of sensing nodes. Consequently, we demonstrate that users' collaboration represents a low cost alternative to improve the system performance in terms of final trace completeness. We also show that the accuracy of a trace depends on the spatial-temporal distribution of the sensing nodes.
- We identify possible weaknesses of the proposed system concerning users' participation: we address two possible attacks based on the insertion of fake traces, i.e. attractive and repulsive attacks.
- We propose a metric to identify malicious users participating of the measurement system: We show that it is possible to detect malicious nodes adopting one of the attacks model we identify in this work. We show that such malicious nodes tend to display different characteristics from the other nodes in the accuracy graph we build.

7.1. Detecting vulnerabilities

In this work, we consider two different malicious behaviours (we call them "attacks" for sake of simplicity) based on the captured trace adulteration:

- **Repulsive attack.** It consists of inserting fake traces with complete sequences of frames, i.e., basing on the sequence counter field, apparently no frames are missed.
- **Attractive attack.** It consists of inserting fake traces with empty sequences of frames, i.e., containing only the first and the last frame of a sequence.

Depending on the type of incentive, a malicious user could find benefit prosecuting one of these attacks. For instance, let us suppose that users who succeed to capture a minimum number of frames are rewarded with micro payments, improved connection quality, or better reputation in the internal social system; in this case, malicious users would adopt a repulsive behavior. We name it repulsive because it is also a way to make the measurement system infer that the measurement quality (we formally define the accuracy in Section 7.1) in a certain area is high and, therefore, is operating normally, thus "repulsing" the wireless infrastructure away. If, on the other hand, users who find leaks in the measurement system are rewarded, they could adopt the second kind of attack that we call attractive. In this case the attacker could make believe that there is the need to move the infrastructure to the area where the attacker operates. Then the infrastructure is "attracted" toward that geographical area. This same attack could be also convenient to give the impression that the network is

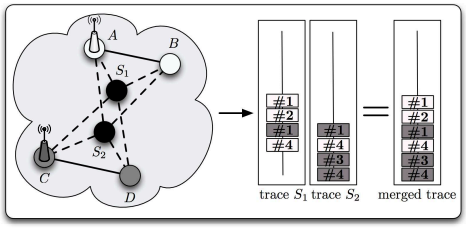


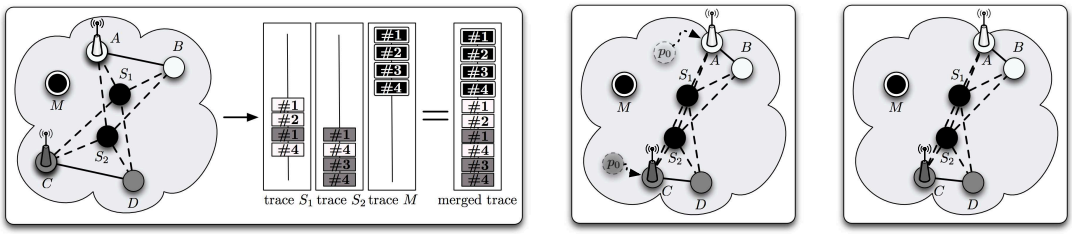
Figure 7.1: The fraction of captured frames increases after merging two individual traces from different sensing nodes. Each individual sensing node, S_1 and S_2, has captured 50% of the total frames numbered in sequential order from 1 to 4. After merging, the fraction of captured frames increases to 75%.

overloaded in an area with the aim to attract more network resources and benefit from the improved quality of their own wireless access network.

We would like to underline that, independently of the attack, this could be motivated by the possibility of simply disrupting the network operation. In a corporate building, for instance, any attack could be triggered by an intruder only motivated to start a counterproductive action. Alternatively, a malicious user logged in to a private or corporate environment could use the attractive attack to draw the wireless infrastructure in places where it is insufficient, i.e., at the far ends of a building.

Figure 7.1 illustrates the normal operation of the system. In this figure, two sensing nodes, S_1 and S_2, overhear different areas within the same wireless network. These nodes are part of the sensing module, where each one can be dedicated or from a user contributing to the monitoring system. Nodes A and C, on the other hand, are wireless infrastructure nodes, while B and D are simple communicating nodes. In the figure, the dotted lines represent the links between the sensing and the communicating nodes, whereas the full lines represent the links between the communicating nodes. Note that the sensing node S_1 captures more frames from the communication between nodes A and B (white frames), whereas S_2 captures more frames from C and D (gray frames). Assuming that on each communication, nodes exchange four frames as enumerated in the figure, nodes S_1 and S_2 capture 50% of the total number of transmitted frames. $S1$ captured $3/4$ of the frames from AB and $1/4$ from CD, whereas S_2 captured $1/4$ from the AB and $3/4$ from CD. After merging these two traces, the final result has 75% of the total frames, which leads the system to improve its performance.

Figures 7.2 and 7.3 illustrate the repulsive and the attractive attack, respectively. We now have a malicious node M, under control of a malicious user, inserting a fake trace in the system. In a repulsive attack, as illustrated in Figure 7.2(a), the malicious node M forges a trace with all four frames from a nonexistent communication (i.e., it creates from scratch a pair of communicating nodes). After merging, the fraction of captured frames over the total is approximately 83%, which is higher than the 75% captured in normal operation (Figure 7.1). As a possible attack outcome, the available infrastructure (nodes A and C) can

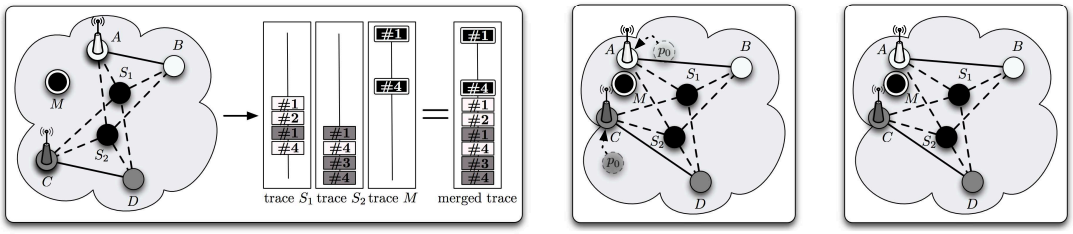


(a) Malicious node M inserts fake traces with complete sequences of frames.

(b) Infrastructure nodes in their initial position p_0

(c) Network final position. move farther way from M.

Figure 7.2: Repulsive attack. Malicious node M forges a trace containing a complete sequence of frames. As a consequence, the fraction of captured frames increases from 75% to 83% and the wireless infrastructure is repulsed to other areas. At the end, the wireless infrastructure becomes farther away from M.



(a) Malicious node M inserts fake traces with empty sequences of frames.

(b) Infrastructure nodes in their initial position p_0

(c) Network final position. move closer to M.

Figure 7.3: Attractive attack. Malicious node M forges a trace containing an empty sequence of frames. As a consequence, the fraction of captured frames reduces from 75% to 66% and the wireless infrastructure is attracted toward M. At the end, the wireless infrastructure becomes closer to M.

be moved to an overloaded region as seen in Figures 7.2(b) and 7.2(c). On the other hand, in the attractive attack (Figure 7.3(a)), after the merge, 66% of the frames are considered to be captured. Thus, the fraction of the total frames becomes lower than the one obtained by the normal operation. The consequence, could be the attraction of more infrastructure resources (nodes A and C) towards the malicious node.

We assume that each sensing node contributes with only one trace. This is an important assumption because the impact of a malicious node can be accentuated by the number of traces it adds to the system. In addition, each trace has the same format, is obtained or forged during the same time frame, and contains coherent amount of data. We consider that the merging software discards traces which do not match these assumptions.

7.1.1. Accuracy

The goal to conduct collaborative measurements is to improve traffic characterization and, consequently, avoid misunderstandings or erroneous actions based on biased measurements. It is of utmost importance to have accurate traffic measurements for the sake of positioning wireless infrastructure. Therefore, we formally define accuracy later on in this section, based on three main assumptions:

1. Each node in the network sends a sequence of frames sorted by an increasing sequence number to any other node in the network.

2. Although measures are performed in a distributed fashion, traces are merged in a central point. We consider that such procedure is CPU- and time-consuming and, therefore, is better handled in a central point. Multihop transmissions could be required to gather all the traces.

3. There is a non-negligible probability of not recording a frame, even considering all the produced traces.

Based on the above assumptions, we can estimate the maximum number of frames a node can send to a destination as the difference between the maximum sequence number and the first one found in the trace plus the number of retransmitted frames. Since in real experiments it is very difficult to have a complete trace (losses are very frequent), we rely on this metric to estimate the completeness of a captured trace, as it has been done in the literature[98;116]. The IEEE 802.11 standard states that the difference between the sequence numbers of successive frames that are coming from a wireless node should differ by one modulo 4096. Hence, let $\mathcal{N}$ be the set of nodes in the network and i and j two nodes in $\mathcal{N}$. Let $v_i(t)$ denote the set of nodes within the neighborhood i in the time frame t. Hence, we consider that consecutive frames have its sequence number incremented by one and that a retransmission can be estimated by detecting repeated sequence numbers or by detecting a retransmission flag on. The maximum number of frames a node i can send in t, $s_i(t)$, is then:

$$s_i(t) = \sum_{j=0}^{|v_i(t)|} (n_{ij}^{\max}(t) + s'_{ij}(t)), \tag{7.1}$$

where $n_{ij}^{\max}(t)$ and $s'_{ij}(t)$ is the maximum sequence number found from i to j and the number of retransmitted frames also from i to j, respectively. Note that a node i may not send a frame to a given neighbor. In this case, $n_{ij}^{\max}(t) = 0$ and $s'_{ij}(t) = 0$. In addition, as frames can be lost because of physical medium issues, the total number of frames received from node i in t, $r_i(t)$ is upper-bounded by $s_i(t)$ ($r_i(t) \leq s_i(t)$). Computing the number of missed frames

from a node i is straightforward and is obtained by subtracting the maximum number of frames sent from the number of received ones. Hence,

$$m_i(t) = s_i(t) - r_i(t). \tag{7.2}$$

Generalizing Equation 7.2 for all nodes in $\mathcal{N}$, we have that the number of frames missed in t ($m(t)$) is equal to the number of frames sent by all nodes ($s(t)$) subtracted from the number of frames received by all nodes ($r(t)$). Thus, $m(t)$ is computed as follows:

$$m(t) = s(t) - r(t) = \sum_{i=0}^{|\mathcal{N}|}(s_i(t) - r_i(t)), \tag{7.3}$$

where $|\mathcal{N}|$ is the number of nodes in $\mathcal{N}$.

Based on Equation 7.3, we can formally define the accuracy metric.

Definition 1 (Accuracy metric a:)**.** *The percentage of frames sent in the network that was captured by at least one sensing node. Hence,*

$$a = 1 - \frac{m(t)}{s(t)}. \tag{7.4}$$

Accuracy computation requires reading each trace only once. The last sequence number of each frame is compared with the previous one (within the same flow) to check if there are any missed frames.

7.1.2. Detection system

We model the system as a fully connected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote, respectively, the set of vertices and edges. Each vertex T_i represents an obtained trace, whereas each edge represents the accuracy considering the vertices connected. In our model, the graph $\mathcal{G}$ is weighted and the weight assigned to an edge T_iT_j is the accuracy computed considering only these two traces. The graph is considered fully connected because we compute the accuracy for all pairwise combinations. Let $a(T_iT_j)$ be the accuracy computed with respect to the merged trace using as input the pair of traces T_i and T_j, then $w(T_iT_j) = a(T_iT_j)$. Based on the link weight defined, we define the node strength of each vertex ($\sigma(T_i)$) as follows:

$$\sigma(T_i) = \sum_{j=1}^{|\mathcal{V}|} a(T_iT_j), \tag{7.5}$$



(a) Collocated scenario.

(b) Wireless traffic characterization.

Figure 7.4: Experimental scenario.

where the number of traces is equivalent to the number of sensing nodes (i.e., $|\mathcal{V}| = |\mathcal{M}|$). In case of large wireless community networks, we can reduce computation complexity by relying either on parallel computation of node strengths or on a subset of trusted nodes that would be in charge of running the measures. We argue that the weights of the edges containing a malicious trace, generated to trigger one of the attacks, have a discrepant value compared with all the real ones (if the amount of fake information introduced into the system is sufficiently large). This is because the addition of traces with fake communicating pairs has as the effect the accuracy increase or decrease, depending on the attack. The trace with strength different from the others can be detected with a certain probability using an outlier test.

We do not consider either attacks than those described in Section 5.3.2; thus, we cannot guarantee that our detection system would work under other attack models. We believe, however, that attractive and repulsive attacks are the strongest attacks one can make in a collaborative measurement system; this is the reason we focus on them.

7.2. Experimental Setup

We evaluate our approach and investigate the impact of the number of nodes and the distance between sensing nodes, in two scenarios.

- **Collocated Scenario:** the first one, called "collocated", was built inside a room within LIP6 computer science laboratory from UPMC Sorbonne Universités in Paris. All sensing nodes were positioned side-by-side on a table in a room of the LIP6 lab, as illustrated in Figure 7.4(a). Monitors capture the wireless traffic for 90 minutes and individual traces have an average size of 253 MBytes, whereas the merged trace has a size of 450 MBytes. The wireless traffic collected is composed of control, management, and data frames to and from access points in the area. Figure 7.4(b) shows the amount of traffic captured. The traffic characterization is obtained after merging the traces col-

lected by the eight laptops in the same time interval. We plot all the traffic captured on each IEEE 802.11b/g non-overlapping channels, i.e., channels 1, 6, and 11. Note that the network activity is similar in all non-overlapping channels on each scenario. In addition, the overall number of frames collected is higher in the collocated scenario, which can be a consequence of the sensing nodes proximity or simply an occasional characteristic of the measurements. In this work, we decided to use channel 1 as our reference. Nevertheless, it is worth mentioning that the obtained results are not significantly affected by the operating channel.

- **Scattered Scenario:** in contrast to the previous scenario, we make the same evaluations using the IRCICA scenario presented in Section 6.2, where nodes were scattered in the second floor of the IRCICA/LIFL building, as illustrated in Figure 6.3(a).

Although these scenarios are both indoor, they show completely different natures that allow better understanding the operation of a malicious node. We underline that, in both experimentations, sensing nodes capture any transmitted frames they hear in the area (nodes are in monitor mode). This means that the captured traffic can be from an access point within the scenario, but it can also be from a pedestrian carrying on a Wi-Fi mobile phone on a nearby street. We assume that any incoming traffic must be captured by the measuring system, no matter how long it lasts or what kind of activity it is concerned with.

Software tools, to merge and analyse traces, are the same presented in Section 6.2.2.

7.3. Accuracy Measurements

Before addressing the security trends in Section 7.4, let us first investigate the behavior of the measurement accuracy under normal operation of the network.

7.3.1. Individual and merged traces

Table 7.1 shows the accuracy of each individual trace compared with the merged one.[2] The parameters required to compute the fraction of received frames, i.e. the maximum sequence number and the number of retransmissions per source-destination pair ($s(t)$), are extracted from the merged trace. Hence, we say that we have computed the accuracy of all traces using the merged one as the reference. The accuracy difference between the merged trace and the trace with the smallest accuracy varies between $2\times$ in the collocated scenario and $25\times$ in the scattered scenario, considering the lowest accuracy found with individual traces in both cases.

We observe the impact on accuracy of the distance among the sensing nodes even if the metric is always considerably less than one because of the promiscuous nature of the measurements. We do not filter traces, so we considering also the furthest long conversations.

[2]We refer to the "merged trace" as the result of merging *all* individual traces in one.

Table 7.1: Accuracy results obtained with individual and merged traces in both evaluated scenarios.

Scenarios	**Traces**								
	T1	T2	T3	T4	T5	T6	T7	T8	**Merged**
Collocated	0.00157	0.00158	0.00170	0.00157	0.00199	0.00182	0.00237	0.00171	**0.00327**
Scattered	0.00034	0.00029	0.00025	0.00135	0.00101	0.00065	0.00017	0.00012	**0.00307**

Monitors are able to capture only few frames of such conversations. Closer the nodes, the smaller the difference in accuracy between individual and merged traces. This happens because as we scatter the sensing nodes throughout an area, the difference between the individual measured traces becomes more relevant.

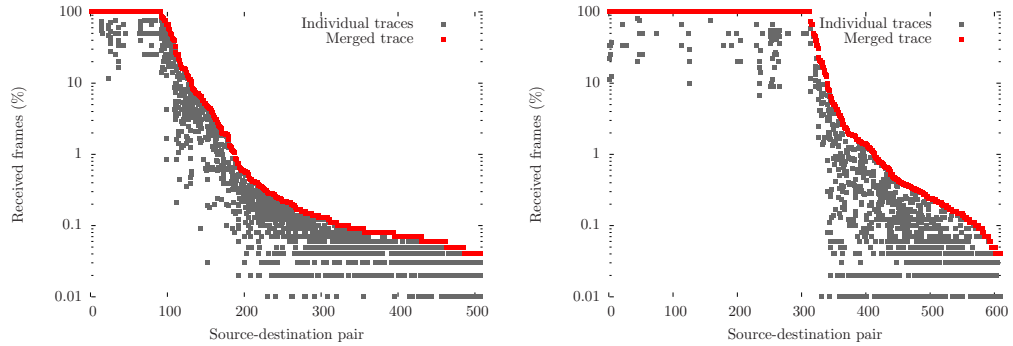
In the next two sections, we compute the fraction of received frames per source/destination pair instead of the accuracy metric. We aim at better observing the impact of the number of traces and of the sensing nodes' position on the collaborative measurements.

7.3.2. Number of traces

Figure 7.5 illustrates the fraction of received frames from all source-destination pairs in the network. We compare the result of the individual traces with the merged one. In the x-axis, we sort the source-destination pairs according to the fraction of received frames. It is important to observe that, in both scenarios, the discrepancy between the fraction of frames received as shown by the merged trace and by a certain individual trace can be very large for a given source-destination pair. Hence, decisions based on a single trace can lead to considerable erroneous actions.

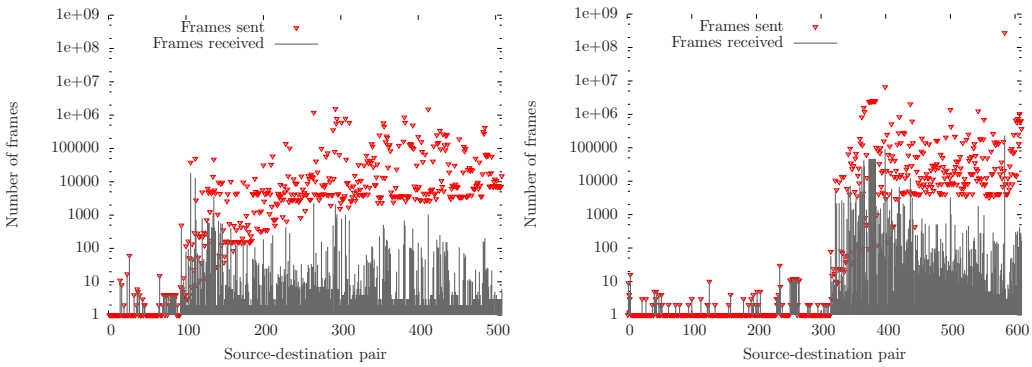
Comparing the results obtained for the collocated and scattered scenarios, we verify a similar behavior and the presence of a plateau in 100%. This is concerned with management as well as small sequences of frames. Management frames are more robust against physical medium issues since they are typically transmitted at the network base rate. Small sequences of frames, on the other hand, can be totally captured if at least one sensing node overhears the transmission for a sufficiently long time.

Figure 7.6 shows the difference between the number of frames sent and the number of frames received by each source-destination pair in the network. In this figure, we consider only the merged trace. The x-axis is the same as in Figure 7.5. Again, note that the number of sent and received frames is similar up to a given pair of nodes. This matches the same pair of nodes where the percentage of received frames drops down from 100% in Figure 7.5, i.e., the 100-th pair in the collocated scenario and the 310-th pair in the scattered one. In addition, we observe that the sequence number is small up to the 100-th and 310-th pairs from the collocated and scattered scenarios, respectively. This corroborates our claim that these frames are either management frames or are simply small sequences of frames. There-



(a) Collocated scenario. (b) Scattered scenario.

Figure 7.5: Fraction of received frames considering all source-destination pairs in both scenarios.



(a) Collocated scenario. (b) Scattered scenario.

Figure 7.6: Sequence number variation considering all source-destination pairs in both scenarios.

fore, losses become significant when larger sequences of frames are transmitted at higher transmission rates.

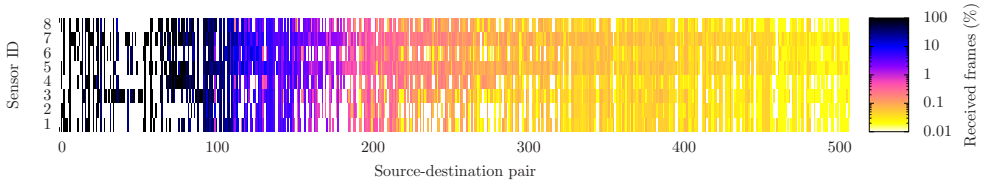
7.3.3. Position of sensing nodes

Figure 7.7 plots the correlation between the fraction of received frames and the sensing node position. In this figure, we plot the accuracy of each individual trace. The y-axis shows the sensor ID, whereas the x-axis is the same used in the previous figures. The sensor IDs correspond to the ones presented in Figure 6.3. As a third magnitude in the plot, we have color intensity to represent the fraction of received frames measured by each sensing node. Again, we use the merged trace as the reference.

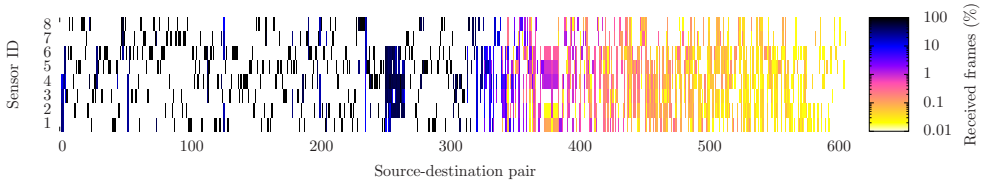
Figures 7.7(a) and 7.7(b) show that up to the 100-th and 310-th pair, respectively, there is always at least one node receiving the transmitted frames. This can be checked by the black points in the plot. Nevertheless, even assuming that these frames are for management, they are frequently not overheard by all sensing nodes. This is because their position can even affect the frames transmitted at lower transmission rates. On the other hand, as the sequence number increases, the percentage of received frames drops for all sensing nodes,

as represented by lighter colors. This indicates that nodes are transmitting at higher rates, as expected for data frames.

An interesting issue from Figure 7.7 is that the fraction of received frames changes according to the sensing node position, independently of the scenario. This observation is done by taking a look at the color variation of any vertical line in the plot, i.e., from any source-destination pair. Figure 7.8 clearer shows this effect. We have sorted in decreasing order the x-axis according to the fraction of received frames of the first sensing node. In case of tie, we compare the fraction of the second sensing node, and so on.

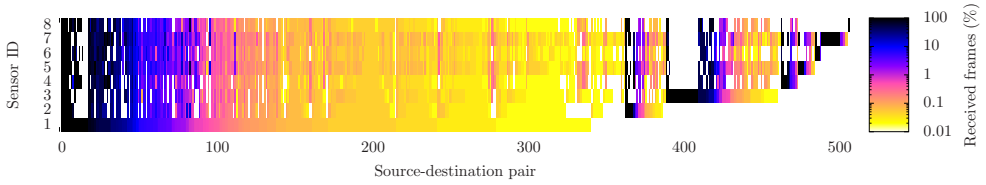


(a) Collocated scenario.

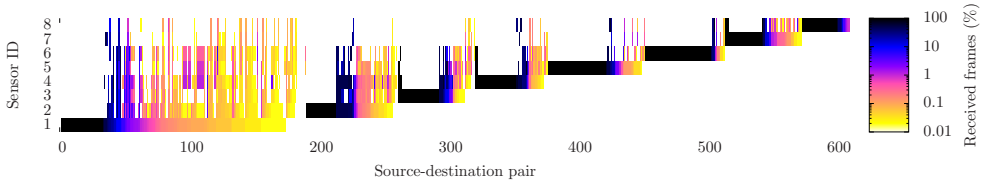


(b) Scattered scenario.

Figure 7.7: Geographical distribution of percentage of received frames considering all source-destination pairs in both scenarios.

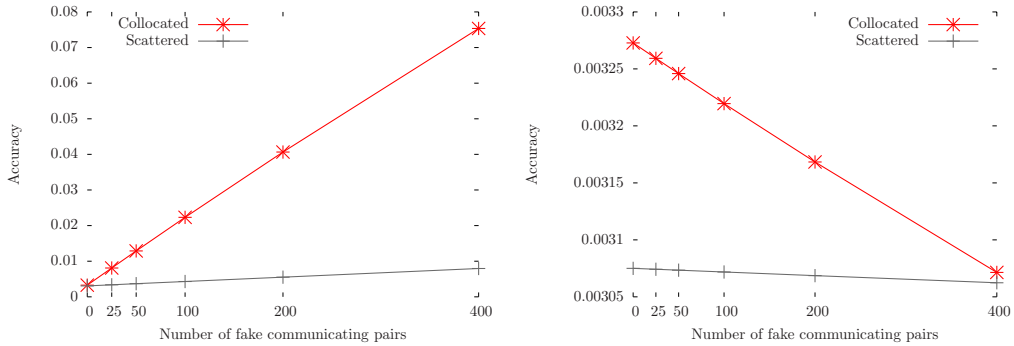


(a) Collocated scenario.



(b) Scattered scenario.

Figure 7.8: Geographical distribution of percentage of received frames for sorted fraction of received frames considering all source-destination pairs in both scenarios.



(a) Repulsive attack.

(b) Attractive attack.

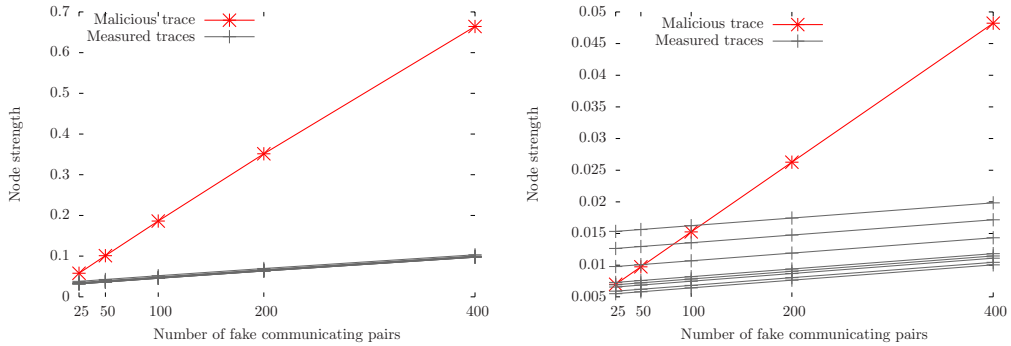
Figure 7.9: Impact of inserting traces with fake communicating pairs on the accuracy of the merged trace.

7.4. Impact of Attacks

In the previous section, we observed that: (i) the accuracy increases with the number of sensing nodes and (ii) even small distances among sensing nodes are enough to change the wireless activity observed by each node. These observations have a direct incidence in our work.

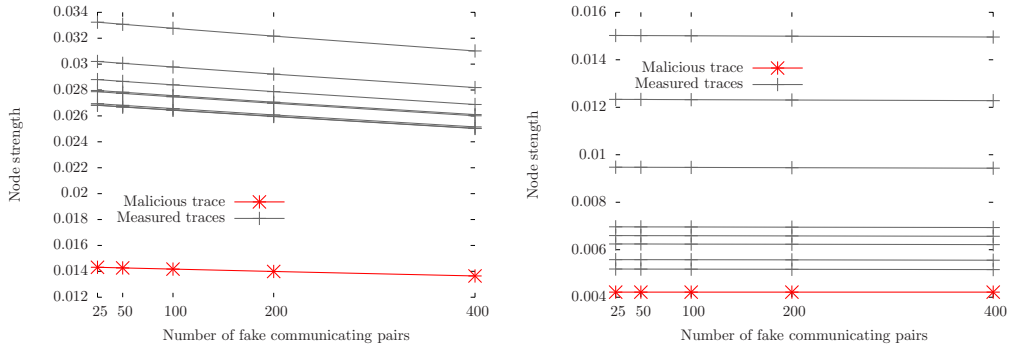
First, increasing the population of sensing nodes also increases the probability that some node from the community be malicious. As stated before, we rely on the accuracy metric to characterize the behavior of sensing nodes. In this section, we evaluate the impact of both attractive and repulsive attacks, described in Section 7.1, on trace accuracy. In both attacks, a malicious user creates from scratch a trace containing fake sequences of frames between imaginary communicating pairs. In the attractive attack, the malicious node creates traces containing only the first and the last frames of a sequence; whereas in the repulsive attack, it creates traces containing complete sequences of frames. The goals of these attacks are to, respectively, artificially reduce and increase the global accuracy of the system.

Figure 7.9 depicts the impact on accuracy of the two attacks in both scenarios. We vary the number of fake communicating pairs added by the malicious node to verify the accuracy variation of the merged trace. Note that attacks in dense scenarios are more effective than in scattered ones. Again, this is because the coherence among the individual traces as collected by closer nodes is higher. Thus, the impact of adding fake communicating pairs is more relevant. Although plots have a similar linear behavior, the repulsive attack is more efficient than the attractive one from the point of view of accuracy change. Whereas in the repulsive attack, the accuracy change is near 2,200% and 150% considering collocated and scattered scenarios, respectively; in the attractive attack, this difference is near 6% and 0.4% also considering the collocated and the scattered scenarios. This is because the accuracy without any attack is already low, as shown in Table 7.1. Therefore, further reducing the accuracy requires a higher number of fake communicating pairs.



(a) Collocated scenario. (b) Scattered scenario.

Figure 7.10: Node strength variation in repulsive attack.



(a) Collocated scenario. (b) Scattered scenario.

Figure 7.11: Node strength variation in attractive attack.

7.5. Detecting Potential Attackers

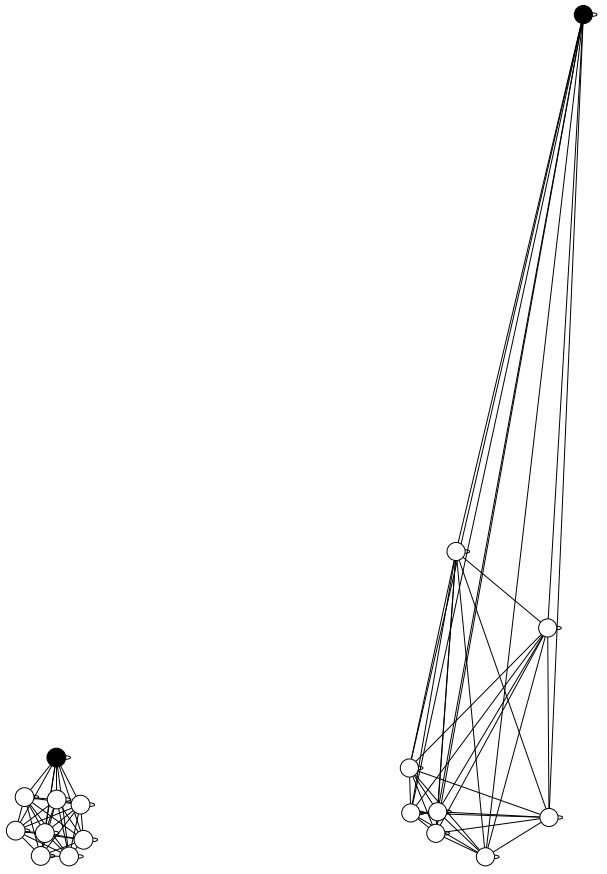
The proposed detection system relies on the accuracy of pairwise merged traces. This means that we first merge all traces two by two and then calculate the accuracy of these merged traces. The goal is to build a graph that will allow us detect nodes with an average different behavior than the others. Figure 7.10 shows the node strength variation of each trace with the number of fake communicating pairs for a repulsive attack (refer to Section 7.1). Note that the node strength of the malicious trace grows up faster in the collocated scenario because the shorter distances among the sensing nodes also contribute to a higher pairwise accuracy. In addition, the node strength difference of the measured traces (legitimate traces) is subtly, which makes them coincide in the plot. Increasing the density of sensing nodes can improve the robustness of the system against the repulsive attack.

Figure 7.11 depicts similar results for the attractive attack. In this case, however, the effect is the opposite in terms of node strength, i.e., the lowest node strength likely comes from a malicious node conducting an attractive attack. Note that the impact is not as evident as in the repulsive attack. In the collocated scenario, there is a reduction in the node strength of all legitimate traces as a consequence of computing their accuracy with the malicious

trace. In the scattered scenario, however, the node strength reduction is barely seen, which shows that this attack requires more fake communicating pairs to be effective. It is worth mentioning that if the impact of the attack is not easily observed, the proposed system will not be able to clearly identify the malicious user. This is not an issue since the efficiency of the detection system is somehow proportional to the impact of the attack.

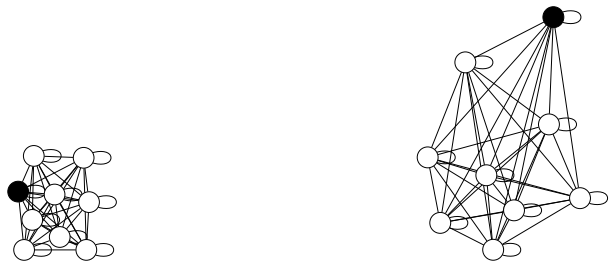
Figures 7.12, 7.13, 7.14, and 7.15 plot the graphs adjusting the edge length as a function of its weight. We used the `graphviz` software to accomplish this task[37]. The visualization can also help detecting a malicious trace and, furthermore, a malicious user. In these figures, the black and the white circle represent the malicious and the legitimate trace, respectively.

Figures 7.12 and 7.13 plot the impact of the repulsive attack in collocated and scattered scenarios, respectively. In both figures, the vertex representing the trace from the malicious node becomes more distant from the others as the number of fake communicating pairs is inserted. We only show results from the extreme cases considered, i.e. 25 and 400 fake communicating pairs. Intermediate results show only a progressive behavior and, therefore, are omitted. In the collocated scenario, however, the difference is more evident for a fewer number of communicating pairs because the pairwise accuracy in this scenario is higher.



(a) (b)

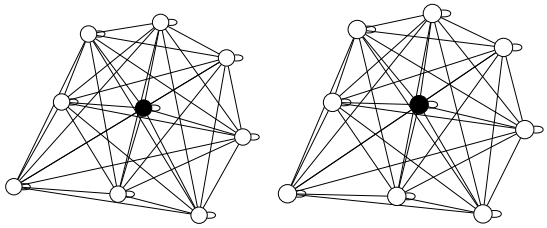
Figure 7.12: Graph obtained considering a malicious node executing the repulsive attack in the collocated scenario: (a) with 25 fake communicating pairs and (b) with 400 fake communicating pairs.



(a) (b)

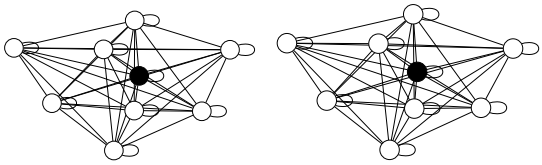
Figure 7.13: Graph obtained considering a malicious node executing the repulsive attack in the scattered scenario: (a) with 25 fake communicating pairs and (b) with 400 fake communicating pairs.

Figures 7.14 and 7.15 plot the impact of the attractive attack in collocated and scattered scenarios, respectively. Because the attractive attack aims at reducing the measurements accuracy, all the edge weights containing a malicious trace tend to reduce. Thus, in opposition to the repulsive attack, the malicious node becomes located in a central position in the graph. Again, the effect of the attack is more evident in the collocated scenario.



(a) (b)

Figure 7.14: Graph obtained considering a malicious node executing the attractive attack in the collocated scenario: (a) with 25 fake communicating pairs and (b) with 400 fake communicating pairs.



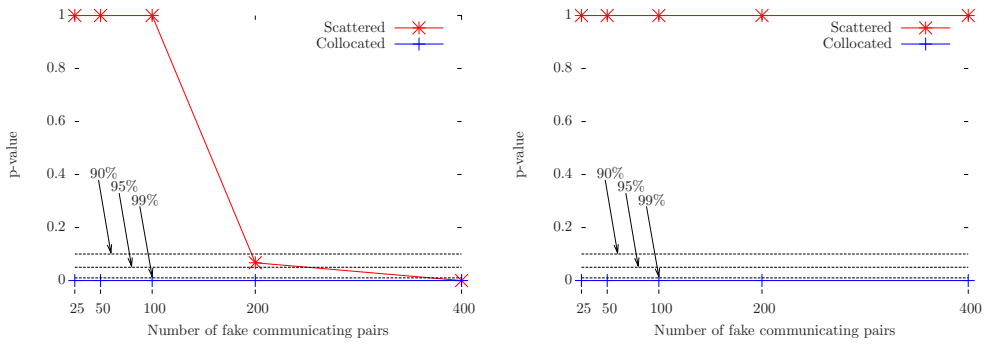
(a) (b)

Figure 7.15: Graph obtained considering a malicious node executing the attractive attack in the scattered scenario: (a) with 25 fake communicating pairs and (b) with 400 fake communicating pairs.

We also conduct an outlier test over the node strengths computed for the two attacks in both scenarios. We use Dixon's test because it can be run over small data sets[30]. The goal is to demonstrate the possibility of detecting a potential malicious trace using simple tests. Thus, we evaluate the hypothesis of a given node strength be an outlier or not. To this end, we first verify if the node strength follows a normal distribution. Running Cramér-von Mises test[29] for all node strengths of the legitimate traces, we could not reject the

Table 7.2: Cramér-von Mises results for normality hypothesis test. The results above 0.05 do not reject the hypothesis of normality.

Scenarios	Legitimate	Repulsive Attack	Attractive Attack
Collocated	0.09199	4.224×10^{-7}	0.0006139
Scattered	0.08413	0.005008	0.08194



(a) Repulsive attack. (b) Attractive attack.

Figure 7.16: Outlier detection using Dixon's test.

hypothesis of following a normal distribution. In both cases, the p-value is above 0.05, which gives a confidence level of 95%. In our model, all nodes have the same degree and the node strength becomes a function of pairwise accuracy.

Note that a preliminary test could be conducted simply verifying the degree distribution. Hence, if the distribution deviates from normality, this can represent a potential attack. Table 7.2 presents the p-values obtained with the Crameér-von Mises test, considering the node strength computed for the traces without attack, i.e., with only the eight legitimate traces; and with each attack, i.e., with all the eight legitimate traces plus the malicious one. We show the results obtained with 400 forged communicating pairs. Note that we can have already a hint that an attack is under way since the hypothesis of following a normal distribution could not be rejected. In the scattered scenario with attractive attack, we cannot have such idea since the scenario does not lead to enough difference among node strengths.

Figure 7.16 shows the result of the Dixon's test considering 90%, 95%, and 99% confidence levels. Similarly to the normality test, we evaluate the hypothesis of the malicious trace be not an outlier. We observe that the hypothesis is never confirmed for the collocated scenario, independent of the attack. On the other hand, in the scattered scenario, the hypothesis is only not confirmed in the repulsive attack for more than 200 and 400 malicious communication nodes – the former with 90% of confidence level, whereas the latter with more than 99%.

7.6. Summary

We prove that the geographical distribution as well as the number of collected traces can significantly improve the correctness of the merged trace, allowing better informed decisions. To increase the number of traces, we rely on users participation, which can extract benefits from more accurate monitoring. The shortcoming of users' participation is, however, the possibility of malicious actions.

In this chapter, we have experimentally evaluated the impact of malicious nodes in distributed collaborative measurements in wireless community networks. We identify two possible attacks: repulsive and attractive. The first one is based on the insertion of fake traces with complete sequences of frames, whereas the latter is based on the insertion of sequences of frames containing only the first and the last frames. As a consequence, users can, respectively, repulse infrastructure as an attempt to provoke a denial of service attack or can attract more infrastructure towards its location. We have shown that the repulsive attack has more chances to succeed, especially in scenarios where sensing nodes are close to each other.

We have also proposed a detecting system which models the system as a fully connected graph, where each vertex is a trace from a sensing node and each link represents the accuracy of the connected traces. From this model, we were able to infer with certain probability a malicious node by computing the node strength of each trace. Our detecting system is based on the analysis of possible outliers since the node strength computed for malicious nodes tends to be discrepant when compared with legitimate traces. We believe the detection system output can be even used as a trust metric for user behavior evaluation. As a future work, we plan to conduct more experiments and to formal define a trust metric.

Chapter 8

Conclusion and perspectives

8.1. Conclusion

The opportunistic communication mode has several advantages. It is useful for communication and content dissemination purposes in contexts where there is no network infrastructure. When, at the contrary, an infrastructure is present, the opportunistic content dissemination gives benefits offloading the infrastructure. It is also important in environments where nodes are highly mobile and their contacts are very short.

In this thesis we focused on the fair opportunistic dissemination of multiple large contents. We implemented EPICS, an opportunistic content dissemination protocol atop PePiT, an Android-based application running on off-the-shelf handsets. By running PePiT on a real setup composed of several smartphones, we could test the goodness and fairness of EPICS. At the same time, leading experiments on real devices in uncontrolled environments, we discovered some limitations for the content dissemination due to the protocol design itself. Other results were unexpected.

For this reason, we started to capture the wireless traffic generated by our devices along with the surrounding traffic. With a more complete view, we had some insights to improve the diffusion latency. We designed then DAD, a solution that extends EPICS to dynamically adapt the amount of pieces to send according to the density of the network. With respect to EPICS, DAD improves the diffusion latency when the node degree is less than 5. In order to study the profit margin of this improvement, we analyzed real-world and synthetic mobility traces and showed that, depending on the node density and detecting range, DAD has different levels of impact. As a matter of fact, we ran more than 500 experiments (equivalent to about 300 hours), collected and analyzed around four thousand application level logs and 60 Gbytes of wireless traces.

During all the experiments, we captured the wireless traffic using a passive monitoring system. In particular, it helped us design DAD and confirm that EPICS limitations were due only to the protocol design, and not to surrounding (uncontrolled) traffic. As a consequence, with the purpose to deploy a large testbed with much more mobile nodes, new challenges

about how to monitor traffic in a large area arises. The second part of this thesis was centered on this issue. We made two main contributions.

The first one is based on trace similarity and community detection algorithms. Such an approach allows selecting only a significant subset of traces to merge. Monitors producing the other traces could be either suppressed or relocate to enlarge the area under monitoring. In this way, having a fleet of monitors, we can detect the widest area that is possible to monitor without impacting the capture quality. Similarly, given an area of interest, we can detect the number of monitors to employ for a high ratio between quality and costs.

The second approach is a collaborative wireless measurement system where users contribute to the monitoring in exchange of something (e.g., connectivity). Being totally decentralize, this method could suffer from malicious users' activities. For this reason, we also introduced two kinds of possible attacks and relative countermeasure to detect such attacks. We have tested our methods in three scenarios in distinct locations and with different monitors' position.

8.2. Perspectives

Enclosing and combining two topics, this thesis raises several possibilities of future work.

Testbed scalability. The large scale deployment of PePiT is limited by Google's policies forbidding the 802.11 ad hoc communications for Android operating systems. A quite invasive procedure is needed to enable this feature on off-the-shelf devices. At present, PePiT supports a limited number of handset models. To tackle this limitation, we made PePiT to support the Android-x86, a porting of the Android operative system to the x86 platform. In this way, potentially every laptop with an Android-x86 virtual machine can run PePiT. Moreover, the second part of this thesis deals with how to scale a monitoring system to capture wireless traffic during experiments. Anyway testing PePiT in a large scale mobile scenario remains an issue for the amount of people and equipment involved.

Communication technology. When PePiT has been implemented, Bluetooth and 802.11 Wi-Fi were the only possible communication technologies. Some devices were shipped with Wi-Fi Direct, but no APIs were yet available to exploit it. As a future improvement, PePiT may use Wi-Fi Direct or Wi-Fi Opp. This change could make PePiT deployable on every Android device. Different technologies will have also an impact on energy consumption.

More dynamics. DAD modulates the burst of pieces to send in according to the neighborhood size. This adaptation leads to a faster content dissemination and we showed that it is exploitable in many contexts. While the chunk size and the transport protocol have fixed values, other parameters may change accordingly to current network environment (e.g., transmission rate and beaconing frequency).

Content selection. The grey relational analysis (GRA) paradigm provides an extensible inter-content selection strategy. We based our selection on content size and creation time, but many other options are possible. We may give priority to contents with a larger part still to fetch or to contents which are flagged as important. Without, or in collaboration with, the GRA, we may use content diffusion prediction techniques.

Security. During our experiments, we assumed that every node was interested in all contents since we were interested to the content diffusion latency. We did not take into account issues like content integrity, privacy, and security. In order to deploy PePiT beyond our boundaries, we must address the problem of protecting and detecting contents corrupted by untrusted peers.

Monitoring system. Although the weighted inter-flow similarity metric has a good impact on the community detection, many other similarity metrics are possible. Different kinds of community detection algorithms (hierarchical with a divisive approach, modularity-based, etc.) are available too. The TSP solution, as a ranking method, leads to good performance against a ranking solution based on trace size or node degree. We wish to compare it with other ranking methods that are able to sequentially choose the most relevant nodes from different communities.

Trace corruption detection. Since the tests we used to detect outliers among captured traces are statistic, we can use additional metrics to evaluate the obtained traces. For example, we can combine our tests with trust metrics to obtain a final evaluation[113]. The result of the outlier test proposed should be used as an input of the trust metric, which could also consider other inputs such as reputation to avoid premature decisions. In addition, the same framework we proposed, starting from the node strength ($\sigma(T_i)$) computation, could be repeated after removing a malicious trace so as to verify if another could also be detected.

Appendices

Appendix A

Résumé de la thèse en français

A.1. Introduction

La croissance du trafic à partir d'appareils mobiles et sans fil a des prévisions plus que optimistes[3]. Dans ce sens, ainsi que l'introduction de framework innovantes de développement mobile, des nouvelles applications de partage opportunistes du contenu sont surgis[7;16;70;72;75;110].

Ces applications répondent à la demande de l'actuelle société à produire et à consommer des contenus de taille de plus en plus important générés par les utilisateurs (UGC)[23] selon le paradigme triple A (Anyware, Anytime, Any device).

La compatibilité des appareils mobiles sans fil à le paradigme triple-A justifie l'effort prodigué par la communauté de recherche sur les réseaux opportunistes dans les dernières années. Les réseaux opportunistes sont créés par des contacts sporadiques et directes entre les utilisateurs mobiles. Cette particularité les rend aptes à échanger des contenus dans de nombreux contextes et des environnements tels que : les événements locaux et temporaires, de reprise après sinistre et les endroits bondés, les réseaux de véhicules et de capteurs et les communications entre satellites[20;39;43;44;51;76;121]. Ce type de réseau est parfois identifié comme Delay/Disruption Tolerant Networks (DTNs) parce que il est mis en compte un retard pour la réception du contenu.

Comprendre la dynamique au niveau d'application et des mécanismes de la communication sans fil sous-jacent devient fondamentale pour concevoir un protocole d'échange de contenu efficace, en particulier dans des environnements surpeuplés ou dans des situations où de courtes fenêtres de contact sont la règle .

Dans cette thèse, nous abordons ces problèmes en reliant le rendement d'applications de partage opportuniste du contenu avec le trafic sans fil environnante. Sans diminuer l'importance des simulations, seulement dans des conditions très proches à la réalité nous pouvons faire face à des questions pareils. Pour cette raison, nous avons délibérément adopté une approche expérimentale et fondé notre analyse sur des applications réelles déployées sur les dispositifs disponibles sur le marché.

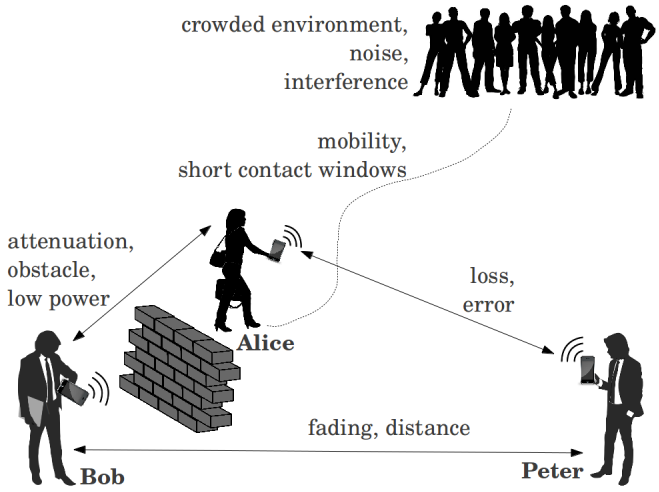


Figure A.1: Problèmes qui se posent dans les communications sans fil.

A.2. Définition du problème

Réseaux Opportunistes. Supposons que Bob veut transmettre un message à Peter par l'aide de communications opportunistes comme montré par la Figure A.1. Ils sont trop loin l'un l'autre pour communiquer directement. Bob pourrait alors envoyer le message à Alice qui passe à coté. Divers problèmes de communication sans fil peuvent se produire: atténuation par un obstacle, multi-chemin, erreurs de transmission, perte de messages pour ne citer que quelques-uns. Si Alice parvient à recevoir le message, elle l'enregistre et le transporte jusqu'à ce qu'elle parviens dans la plage de transmission/réception de Peter. Un tel mécanisme, appelé "store-carry-and-forward" est l'approche générale pour router les messages dans réseaux DTN.

Les protocoles de routage pour les DTN doivent négocier le compromis entre la charge de communication et la performance de livraison. Ce compromis à été étudiée dans nombreux systèmes de routage qui tiennent en compte de la mobilité des nœud et la probabilité des contacts [11;19;45;63;65;104;109;119;120;122].

L'architecture DTN est intrinsèquement centré sur les nœuds avec transmissions directs entre eux indépendamment des protocoles de transport sous-jacents [22]. Les messages peuvent être fragmentés et les fragments peuvent être regroupés n'importe où dans le réseau. Hacher de grandes contenus en petits morceaux pour une diffusion plus efficace est, en fait, une approche naturelle à adopter dans les DTN et c'est l'approche utilisé dans cette thèse aussi. Néanmoins, plusieurs problèmes (étonnamment sous-considérés dans la littérature) se posent:

- Quel est le contenu à transmettre quand un contact se produit?
- Une fois le contenu est sélectionné, quelle pièce devrait être choisi?

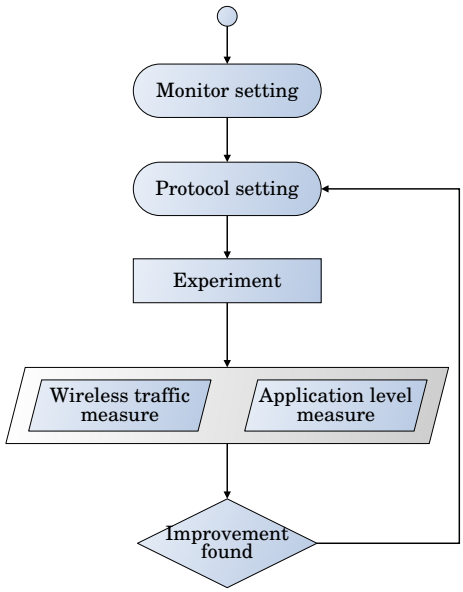


Figure A.2: Flux de travail.

- Quelle taille devrait avoir un morceau?
- Est-il intéressant d'utiliser un protocole de transport fiable?
- Est-il utile de transmettre une rafale de morceaux?

Nous étudions tous ces aspects dans la première partie de ce travail.

Systeme de surveillance. Nous avons trouvé très utile l'analyse de traces de trafic sans fil afin de trouver des réponses à des comportements inattendus et de trouver les limites intrinsèques du protocole réseau dédié au partage de contenu dans DTN. Nous présentons dans la figure A.2 notre flux de travail expérimental complet. Une fois le terrain de l'expérience est choisi, nous procédons au réglage des paramètres du protocole et du système de surveillance passive. Ce dernière concerne de trouver la quantité de moniteurs nécessaires pour avoir une capture qualitative et leur position. Le paramètres du protocole impliquent le réglage de paramètres spécifique à l'essai. De cette façon, nous obtenons deux mesures de niveau application et niveau de liaison de données. Si l'analyse de ces mesures met en évidence des améliorations possibles, les paramètres de protocole sont ajustés en conséquence et nous pouvons commencer une nouvelle série d'expériences.

Un système de surveillance passive est généralement composé de trois modules principaux : détection, fusion et présentation comme dans la Figure A.3. Dans le module de détection, des nœuds de surveillance sont responsables de recueillir les trames qu'il observe dans le medium sans fil.[1] Comme il y a potentiellement plusieurs moniteurs, la sortie est

[1]Nous considérons l'activité sans fil à la couche MAC. Pour cette raison, une trace est un ensemble de trames MAC qu'un moniteur en mode promiscuité peut entendre.

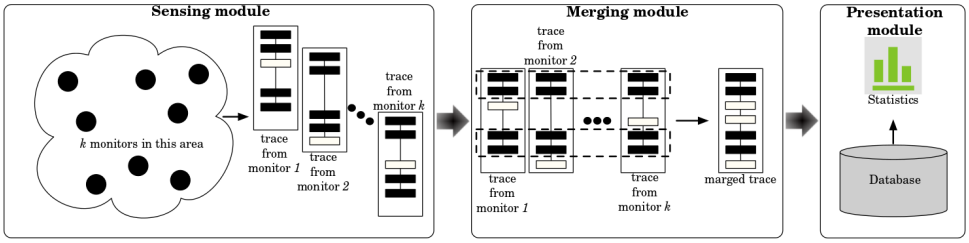


Figure A.3: Modules de détection, de fusion et de présentation : typique architecture du système de surveillance passive.

une collection de traces obtenues par les différents observateurs. Pour simplicité, nous supposons que chaque moniteur ne produit qu'une seule trace. Le module de fusion est en charge d'une trace unique en utilisant comme entrée toutes les traces collectées par le module de détection. Enfin, le module de présentation est responsable de stocker les mesures précédentes et à fournir des statistiques sur l'activité du réseau sans fil.

Les principaux problèmes qui découlent de cette approche sont les suivants :

- Passage à l'échelle du module de détection.
- Placement des moniteurs.
- Complexité de calcul du module de fusion.
- Mesures biaisée en raison de traces fallacieuses .

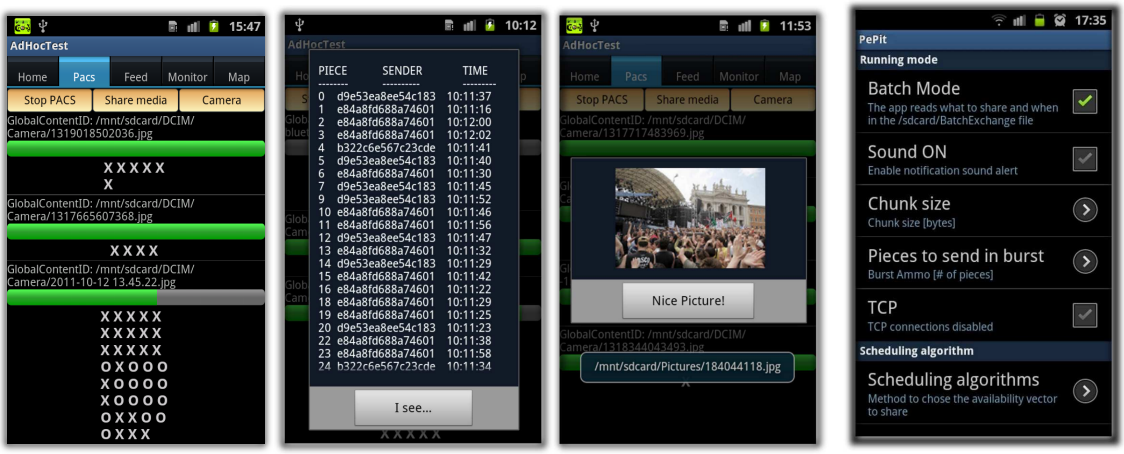
Nous étudions tous ces aspects dans la deuxième partie de ce travail.

A.3. Contribution 1 : PePiT, un substrat basé sur Android pour la diffusion multi-contenu

Malgré le nombre de protocoles et de stratégies de diffusion opportunistes fondées sur la mobilité des utilisateurs et leur comportement social, il n'y a pas autant de véritables mises en œuvre. Pour combler cette lacune, nous avons développé une application mobile pour Android appelé PePiT[95] qui est basée sur le protocole EPICS et montré dans la Figure A.4.

EPICS comprend une stratégie de sélection inter-contenu basée sur l'analyse relationnelle gris[32] pour de choisir, à chaque rencontre entre deux nœuds, le contenu à échanger afin d'avoir une diffusion rapide et équitable entre tous les contenus. Dans notre travail, l'analyse gris relationnelle tend à donner la priorité aux contenus de grosse taille et aux contenus plus récemment créé. Mais EPICS comprennent également PACS, une stratégie de sélection intra-contenu[14;15]. Une fois le contenu à partager est choisi, PACS choisit quelle pièce de ce contenu à transmettre.

A.3.1. Évaluation



(a) PePiT. From left to right: exchanges in progress, history of content pieces received from different peers, preview of the received picture.

(b) PePiT settings menu.

Figure A.4: PePiT user interface.

Le développement de PePiT nous donne l'opportunité d'évaluer EPICS, comme protocole d'échange opportuniste de contenus, dans le monde réel. Nous comparons EPICS à une stratégie uniforme. Nous appelons stratégie uniforme, une stratégie qui, comme EPICS, utilise PACS pour la sélection intra-contenu, mais sélectionne le contenu à transmettre de façon aléatoire uniforme. Expériences durent aussi longtemps que nécessaire pour les deux stratégies pour assurer la pleine diffusion de tous les contenus.

Table A.1: Paramètres expérimentaux with PACS and EPICS.

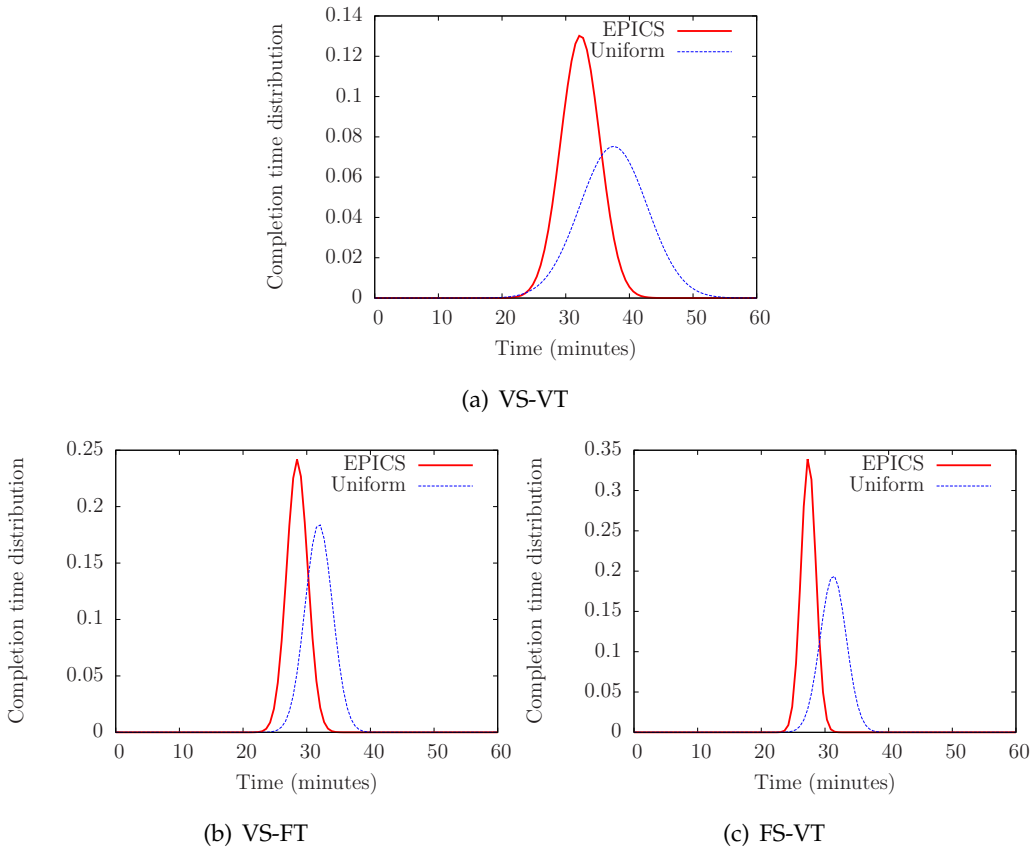
Paramètre	*Valeur*
Protocole de transport	UDP
Nombre de noeuds	10
Nombre de contenus	40
Taille de pièces	64kB
Taille de contenus	16kB, 3.5MB

Nous avons mis 10 téléphones Android (4 HTC Desire et Samsung Galaxy 6-S-II équipé d'Android 2.3.3) sur des emplacements fixes dans un bureau. Les autres paramètres expérimentaux sont montré dans le Tableau A.1.

Nous testons trois scénarios :

- **VS-VT:** taille variable et temps de création la variables. Quarante contenu (dix par nœud) avec des tailles à partir d'un pièce jusqu'à 56 pièces sont créés à des moments différents.
- **VS-FT:** taille variable et temps de création fixe. Les mêmes contenus sont créés simultanément après 2 minutes.
- **FS-VT:** taille fixe et temps de création variables. Quarante contenu de 3 pièces, dix par nœud, sont créées à des instants différents.

Nous répétons chaque scénario dix fois et nous obtenons la durée moyenne pour la complète diffusion μ et l'écart type σ. Avec μ et σ nous construisons les distributions normales



(a) VS-VT

(b) VS-FT

(c) FS-VT

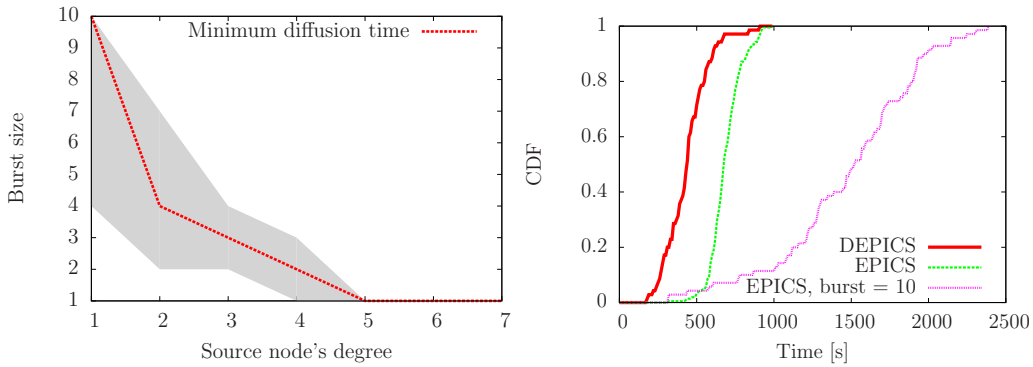
Figure A.5: Distributions des temps d'achèvement des expériences.

représentées dans les figures A.5. Non seulement EPICS est plus rapide que l'uniforme, mais il a aussi un écart plus étroit ce qui signifie qu'il tente de compléter une diffusion équitable entre tous les contenus concernant la taille et l'heure de création.

A.4. Contribution 2 : DAD, EPICS dynamique

Lorsque les nœuds entrent en contact et ils veulent partager des contenus en utilisant EPICS, ils échangent d'abord des informations dans le but de maximiser l'utilité du contact en choisissant la bonne pièce à transmettre. Néanmoins, cet expédient ne permet pas d'éviter totalement pièces doubles car la décision sur le pièce à envoyer est indépendamment prise par chacun des nœuds. Une telle stratégie peut entraîner des frais de communication non négligeable. Une question semble approprié ici : *Serait-il intéressant d'envoyer une rafale de pièces à chaque contact?*

Nous étudions ce point en varier la quantité de nœuds impliqués dans l'échange de contenu. Nous partagions un contenu de 3 Mo avec une seule source et un nœud demandant le contenu (deux nœuds au total), une source et deux autres nœuds(trois nœuds au total), jusqu'à huit nœuds au total, en changent la taille de la rafale de l'un à dix.



(a) DAD zone d'opération : taille de la rafale pour temps de diffusion minimum (ligne rouge) de dix pièces. CDF des temps de dissemination et écart de 30 seconds du minimum (plage gris). par contenu et par noeud.

Figure A.6: DAD: mode d'opération et évaluation.

Dans la figure A.6(a), nous montrons les temps de diffusion minimales recueillies (ligne rouge). La zone grise comprend des valeurs au plus 30 secondes plus que le minimum. Notez que cette zone devient de plus en plus étroite que le nombre de nœuds en contact augmente. Dans ce graphique EPICS se déplace sur le fond, ce qui signifie qu'il peut être amélioré jusqu'à un degré de quatre nœuds. En particulier, dans le cas de deux nœuds (y compris la source), il est intéressant d'envoyer 10 pièces d'affilié, quatre pièces avec trois nœuds, trois pièces avec trois nœuds et deux pièces avec quatre nœuds. D'un nœud degré de cinq haut, il est intéressant de partager un seul morceau par échange par contact.

A.4.1. Évaluation

Nous comparons DAD à la version base d'EPICS (c'est à dire, avec une taille de rafale unitaire) et à la version d'EPICS qui envoie toujours une rafale de dix pièces. La taille de pièce est de 25 Ko. Nous commençons cette expérience avec seulement deux nœuds : une source qui a dix contenu de 3 Mo et un autre nœud. Ensuite, toutes les trois minutes nous ajoutons un nouveau nœud, exigeant tous les contenus, jusqu'à sept nœuds. Nous recueillons des temps relatifs d'achèvement pour chaque contenu pour chaque nœud et nous présentons la CDF dans la figure 4.13. Même si DAD et EPICS prennent le même temps de diffuser tous les contenus sur tous les nœuds, ils présentent une différence considérable jusqu'au 97 centile. Cela signifie que l'adaptation dynamique à la taille du voisinage, non seulement facilite la diffusion quand il ya seulement quelques nœuds, mais, puisque le contenu est presque entièrement reçu dans de nombreux nœuds, ces nœuds peuvent mieux soutenir la diffusion, même quand plusieurs nœuds sont présents dans le réseau. D'autre part, en utilisant EPICS avec la rafale maximum, la diffusion est très rapide pour les premiers contenus (c'est à dire quand il n'y a que quelques nœuds), puis elle ralentit en prenant cinq fois plus longtemps.

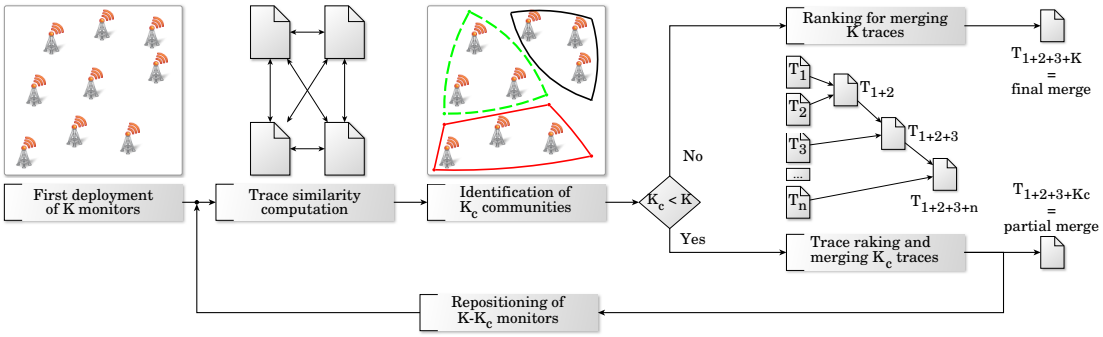


Figure A.7: Tâches supplémentaires proposé pour le module de fusion de traces.

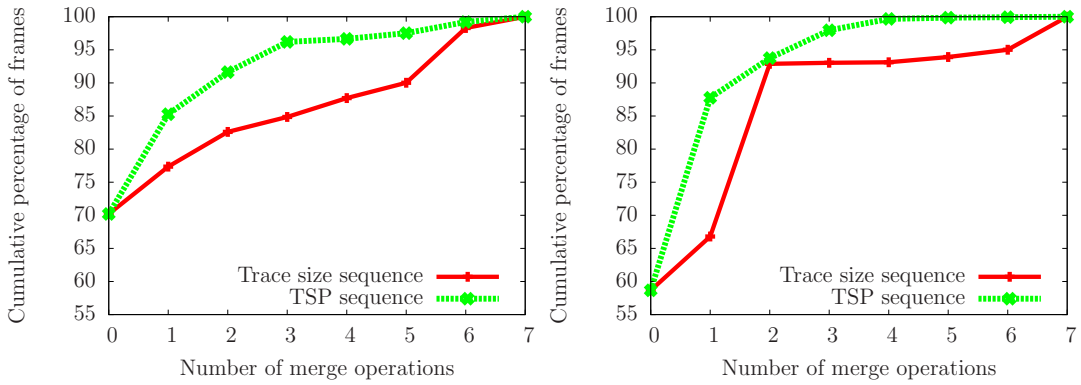
A.5. Contribution 3 : Passage à l'échelle de systèmes de surveillance passive

La première contribution dans le domaine de la surveillance de réseaux sans fil est de concevoir, déployer et tester de nouvelles méthodes de capture du trafic sans fil. Nous essayons de maximiser la quantité de trafic capturé en gardant les coûts. Les coûts sont liés à l'achat, l'installation, le déploiement et la maintenance de moniteurs.

Nous abordons la question suivante : *est-t-il valable de fusionner toutes les traces, ou une partie d'entre eux est suffisant?* L'idée principale derrière notre méthodologie de sélection de trace est montré dans la figure A.7 et repose sur la notion de similitude entre eux (nous testons cinq métriques possibles : *intra-* et *inter-flux*, *Adamic*[6], *Power*[28] et *inter-flux pondéré*[85]), qui peut être utilisé comme entrée d'algorithmes de détection de la communauté (nous avons testé trois algorithmes différents : Walktrap[79] , Infomap[94] et Label Propagation[81]). Ces algorithmes trouver des sous-ensembles de traces avec une similarité élevée (c'est à dire, "une communautés de traces"). Si l'on considère qu'au moins une trace par communauté doit être utilisé dans la procédure de fusion, nous avons une idée du nombre minimum de traces à utiliser. Le problème devient alors de trouver les traces exactes à utiliser dans chaque communauté. Pour cela, nous classons toutes les traces en fonction de leur contribution individuelle à la trace fusionné final, compte tenu des valeurs de similarité par paires[96]. Cette procédure de classement est basé sur le Travelling Salesman Problem (TSP). Nous considérons les traces comme les nœuds d'un graphe et chaque arche entre deux nœuds a un poids proportionnel à la similarité entre eux. La solution du TSP sur ce graphe, en fait, donne le classement.

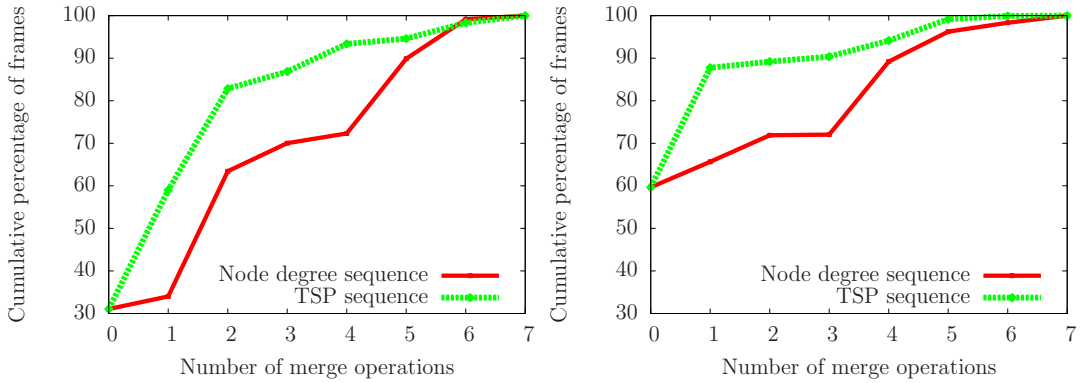
A.5.1. Évaluation

Nous avons fait des expériences dans deux scénarios. Nous appelons le premier "IRCICA", car nous avons déployé huit moniteurs le long un couloir au deuxième étage du laboratoire d'informatique IRCICA de Lille et nous appelons le deuxième "INRIA", car nous avons placé le même huit moniteurs sur l'étage du bâtiment INRIA, à Lille également.



(a) IRCICA scenario. (b) INRIA scenario.

Figure A.8: Comparaison des processus de fusion des traces avec le classement proposé et le classement en fonction de la taille des traces.



(a) IRCICA scenario. (b) INRIA scenario.

Figure A.9: Comparaison des processus de fusion des traces avec le classement proposé et le classement en fonction du degré des nœuds.

Dans les deux cas on compare notre solution de classement de traces avec un classement basé sur le tri décroissante de les tailles des traces (figureA.8) et avec le trie croissante (pareil si décroissante) du degré des traces (figureA.9). Nous considérons le degré d'une trace comme suit: nous considérons les traces comme les nœuds d'un graphe avec les arches entre eux de poids proportionnelle aux valeurs de similarité. Ensuite, le degré d'un nœud est la somme des poids sortant de ce nœud.

Dans tous les cas, notre stratégie, avec un nombre limité de traces (égal au nombre de communautés qui se trouvent sur le graphe) recueille un plus grand nombre de trames, approchant rapidement au montant total de trames capturées. Ca sera donc suffisant, en gardant la qualité de capture, d'utiliser que ces premières moniteurs et, en plus, déplacer les autres pour élargir la surface sous surveillance.

A.6. Contribution 4 : Sensibilité aux traces d'entrée

Nous proposons une méthode collaborative pour améliorer la précision de systèmes de mesure sans fil de réseau. Dans le système proposé, la précision de la trace obtenue est corrélée à l'information spatiale pour donner un indice sur la surcharge des conditions dans un certain endroit. Sur la base de ces informations, il est possible de mieux adapter le réseau en déplaçant des infrastructures supplémentaires pour les zones surchargées.

Dans un système de mesure collaboratif, où les utilisateurs du réseau eux-mêmes contribuent à l'activité de surveillance, le rapport entre le nombre de stations et moniteurs devrait être à la discrétion du concepteur en fonction de ses besoins, en donnant la possibilité l'avoir des solutions peu coûteuses et évolutives. Le problème du placement des moniteurs est résolu d'avantage, parce que nous nous attendons avoir assez de densité dans la plupart des endroits en raison de la prolifération des appareils sans fil[4]. La collaboration des utilisateurs peut également être compensée par des primes, telles que l'allocation des ressources supplémentaires, micro-paiements ou une réputation plus importante basée sur le niveau de contribution.

Cette approche n'est pas exemptée de problématiques. Selon le type d'incitation, un utilisateur malveillant pourrait trouver avantageuse de poursuivre une des attaques suivantes :

- **Attaque répulsive.** Il consiste à insérer des faux traces avec des numéros de séquence complètes (champ sequence number). Apparemment donc il n'y a pas de trames manquantes.
- **Attaque attrayant.** Il consiste à insérer des traces de faux avec que le première et le dernière numéro de séquence.

A.6.1. Système de détection

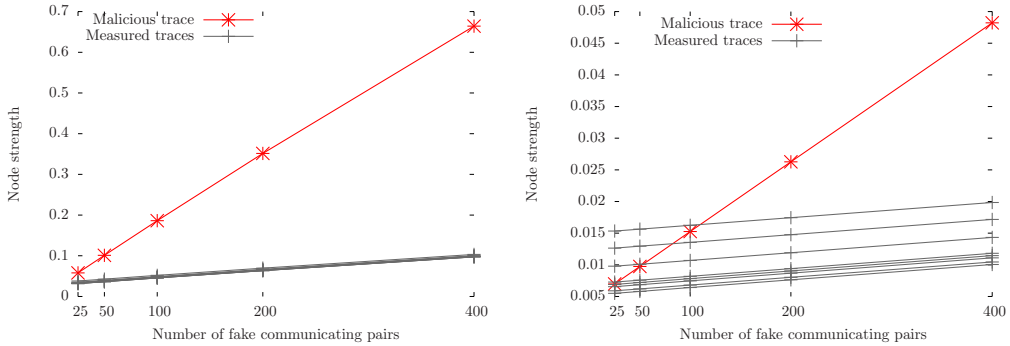
On définit comme *accuracy a* le pourcentage de trames émises dans le réseau qui a été capturée par au moins un moniteurs. Le calcul de l'accuracy nécessite la lecture de chaque trace qu'une seule fois. Le dernier numéro de séquence de chaque trame est comparée à la précédente (du même flux) pour vérifier s'il ya des trames manqués.

Nous modélisons le système comme un graphe entièrement connecté $\mathcal{G}(\mathcal{V}, \mathcal{E})$, où $\mathcal{V}$ et $\mathcal{E}$ désignent, respectivement, l'ensemble des nœuds et des arcs. Chaque nœud T_i représente une trace, alors que chaque arc représente l'accuracy compte tenu des nœuds reliés $a(T_i, T_j)$. On définit le *strenght* d'un nœud, $\sigma(T_i)$, comme:

$$\sigma(T_i) = \sum_{j=1}^{|\mathcal{V}|} a(T_i T_j), \tag{A.1}$$

Table A.2: Accuracy obtenue avec les traces individuelles et avec tous les traces fusionnées.

Scenarios	**Traces**								
	T1	T2	T3	T4	T5	T6	T7	T8	**Merged**
Collocated	0.00157	0.00158	0.00170	0.00157	0.00199	0.00182	0.00237	0.00171	**0.00327**
Scattered	0.00034	0.00029	0.00025	0.00135	0.00101	0.00065	0.00017	0.00012	**0.00307**



(a) Collocated scenario.

(b) Scattered scenario.

Figure A.10: Variation de la métrique strength pour l'attaque répulsive.

A.6.2. Évaluation

Nous effectuons des tests expérimentaux utilisant deux scénarios différents, appelés *scattered* et *collocated*. Le premier est celui qui nous avons appelé IRCICA à la section A.5.1; dans le second, nous mettons les mêmes huit moniteurs très proches les uns des autres sur un bureau.

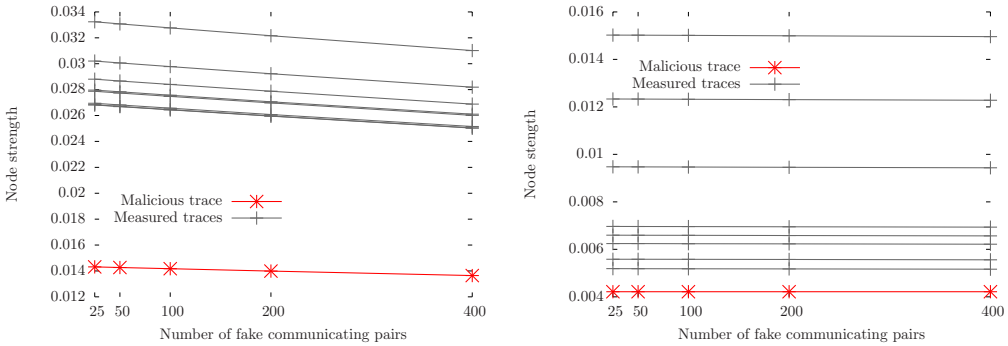
Le tableau A.2 montre l'accuracy par traces individuelles et une fois tous les traces sont fusionnées. Ce dernière valeur est plus important dans tous les deux cas indépendamment de la densité des moniteurs.

Pour tester le système de détection d'attaques, on va ajouter des faux flux dans le traces (25, 50, 100, 200 et 400 flux) et on regarde comment change la métrique strenght pour chaque noued dans les figures A.10 et A.11. Sauf dans le cas d'un attaque attrayant et scenario scattered, le nœud malveillant a une valeure de strength biaisé par rapport à tous les autre nœuds.

Sur cette base, on peut identifier le nœud malveillant soit avec une représentation graphique de $\mathcal{G}$ où les arcs ont une longueur proportionnelle à le strength, soit avec des tests statistiques. Dans ce dernier cas, on utilise le test de Cramér-von Mises (table A.3).

A.7. Conclusions et perspectives

Le mode de communication opportuniste présente plusieurs avantages. Il est utile pour la communication et la diffusion de contenus dans contextes où il n'existe aucune infrastruc-



(a) Collocated scenario.

(b) Scattered scenario.

Figure A.11: Variation de la métrique strength pour l'attaque attrayant.

Table A.3: Test de Cramér-von Mises pour l'hypothèse de normalité. Les résultats au dessus de 0.05 ne rejetent pas l'hypothèse de normalité.

Scenarios	**Legitimate**	**Repulsive Attack**	**Attractive Attack**
Collocated	0.09199	4.224×10^{-7}	0.0006139
Scattered	0.08413	0.005008	0.08194

ture de réseau. Quand, au contraire, une infrastructure est présente, la diffusion de contenu opportuniste donne des avantages pour le déchargement de l'infrastructure. Il est également important dans les environnements où les nœuds sont très mobiles et leurs temps de contacts sont très courts.

Dans cette thèse, nous nous sommes concentrés sur la diffusion opportuniste de plusieurs contenus de grosse taille. Nous avons implémenté EPICS, un protocole de diffusion opportuniste de contenus dans PePiT, une application Android fonctionnant sur portables trouvables sur le marché. Avec PePiT nous avons pu tester la bonté et l'équité de diffusion de EPICS. Dans le même temps, menant des expériences sur des dispositifs réels dans des environnements non contrôlés, nous avons découvert quelques limitations pour la diffusion de contenu en raison de la conception du protocole lui-même. D'autres résultats étaient inattendus.

Pour cette raison, nous avons mis en place la capturer du trafic sans fil généré par nos appareils et du trafic environnant. Avec cette vue plus complète, nous avons eu des idées pour améliorer la latence de diffusion. Nous avons donc conçu DAD, une solution qui étend EPICS en adaptant dynamiquement la quantité de pièces à envoyer en fonction de la densité du réseau. En ce qui concerne EPICS, DAD améliore la latence de la diffusion lorsque le dégrée d'un nœud est inférieure à 5. Afin d'étudier la marge de rentabilité de cette amélioration, nous avons analysé des traces réelles et synthétiques de mobilité et on a montré que, en fonction de la densité de nœuds et de leur plage de détection, DAD a différents niveaux

d'impact. En fait, nous avons ressemblé plus de 500 expériences (équivalant à environ 300 heures), près de quatre mille journaux au niveau application recueillies et 60 Go de traces du trafic sans fil analysées.

Au cours de toutes les expériences, nous avons capturé le trafic sans fil en utilisant un système de surveillance passive. En particulier, ça nous a aidé à concevoir DAD (un nouveau protocole de réseau) et à confirmer que les limitations de EPICS étaient dus uniquement à la conception du protocole, et donc pas au trafic environnant (non contrôlé). En conséquence, dans le but de déployer un grand banc d'essai avec des nœuds mobiles, la complication sur la façon de surveiller le trafic dans une grande région géographique se pose. La deuxième partie de cette thèse a été centré sur cette question. Nous avons fait deux contributions principales.

La première est basée sur la similarité des traces et sur algorithmes de détection de communautés. Une telle approche permet de sélectionner seulement un sous-ensemble significatif de traces à fusionner. Étant donné une plage cible d'intérêt, nous pouvons détecter le nombre minimum de moniteurs à employer pour un grand rapport entre la qualité de capture et les coûts. Les moniteurs produisant les autres traces pourraient être soit supprimées ou délocalisées pour agrandir la surface sous surveillance. De la même manière, ayant une flotte de moniteurs, on peut trouver la surface à surveiller la plus large possible sans nuire à la qualité de la capture.

La deuxième approche est un système de mesure sans fil collaboratif où les utilisateurs contribuent à la surveillance en échange de quelque chose (connectivité, par exemple). Cette méthode, totalement décentraliser, pourrait souffrir d'activités des utilisateurs malveillants. Pour cette raison, nous avons également introduit deux types d'attaques possibles et relatives méthodes pour détecter tels attaques. Nous avons testé nos méthodes dans trois scénarios dans des endroits distincts et avec positions différents de moniteurs.

A.8. Perspectives

Combinant deux thèmes, cette thèse soulève plusieurs possibilités de travaux futurs.

Passer à l'échelle le banc d'essai. Le déploiement à grande échelle de PePiT est limitée par les politiques de Google interdisant les communications 802.11 ad hoc sur les systèmes d'exploitation Android. Une procédure très invasive est nécessaire pour activer cette fonction sur les appareils disponibles sur le marché. À l'heure actuelle, PePiT prend en charge un nombre limité de modèles de téléphones. Pour remédier à cette limitation, nous avons développé PePiT pour être compatible avec Android-x86, un portage d'Android pour les plate-formes x86. De cette façon, potentiellement chaque ordinateur portable avec une machine virtuelle Android-x86 peut exécuter PePiT. Par ailleurs, la deuxième partie de cette thèse traite de la manière de passer à l'échelle un système de surveillance pour capturer le

trafic sans fil au cours des expériences. Néanmoins, tester PePiT dans un scénario mobiles à grande échelle reste un problème pour le nombre de personnes et de l'équipement en cause.

Technologie de communication. Lorsque PePiT a été mis en œuvre, Bluetooth et Wi-Fi 802.11 étaient les seules technologies de communication possibles. Certains appareils ont été livrés avec une connexion Wi-Fi Direct, mais aucune API étaient encore disponibles pour l'exploiter. Comme amélioration future, PePiT pourrait utiliser le Wi-Fi Direct ou Wi-Fi Opp. Ce changement pourrait rendre PePiT compatible avec tous les appareils Android. Différentes technologies auront aussi un différent impact sur la consommation d'énergie .

Dynamique. DAD module la rafale de pièces à envoyer en fonction de la taille du voisinage. Cette adaptation conduit à une diffusion du contenu plus rapide et nous a montré qu'il est exploitable dans de nombreux contextes. Tandis que la taille de pièces et le protocole de transport ont des valeurs fixes, d'autres paramètres peuvent être modifiés en conséquence à l'environnement réseau courant (par exemple, le taux de transmission et la fréquence de balisage) .

Sélection du contenu. Le "grey relational analysis" (GRA) fournit une stratégie extensible de sélection entre plusieurs contenus. Nous avons basé notre sélection sur la taille du contenu et le temps de création, mais de nombreuses autres options sont possibles. Nous pouvons donner la priorité au contenu ayant la plus grande partie encore à chercher ou à des contenus qui sont marqués comme importants. Sans, ou en collaboration avec, le GRA, nous pouvons utiliser aussi des techniques de prédiction de diffusion du contenu.

Sécurité. Au cours de nos expériences, nous avons supposé que chaque nœud est intéressé à tous les contenus car nous nous sommes intéressés à la latence de diffusion. Nous n'avons pas pris en compte des questions comme l'intégrité du contenu, la confidentialité et la sécurité. Pour rendre PePiT disponible au-delà de nos frontières, nous devons aborder le problème de la protection et de détection de contenus corrompu par utilisateurs malveillants.

Système de surveillance. Nombreuses métriques de similarité entre traces sont possibles au delà de celles proposées. Différents types d'algorithmes de détection de communautés (hiérarchiques avec une approche de division, basée sur la modularité, etc.) sont également disponibles. La solution TSP, comme méthode de classement, conduit à une bonne performance contre une solution de classement basé sur la taille des traces ou le degré des nœuds. Nous tenons à le comparer avec d'autres méthodes de classement qui sont en mesure de choisir les nœuds les plus pertinentes provenant de différentes communautés.

Détection de traces fallacieuses. En plus de méthodes statistiques que nous avons utilisés pour détecter des valeurs aberrantes parmi les traces capturées, nous pouvons utiliser des paramètres supplémentaires pour évaluer les traces obtenues. Par exemple, nous pouvons combiner nos tests avec des mesures de confiance afin d'obtenir une évaluation finale[113]. Le résultat du test de valeurs aberrantes proposé devrait être utilisé comme entrée de la

métrique de confiance, ce qui pourrait aussi envisager d'autres inputs, tels que la réputation, au fin d'éviter des décisions prématurées. En outre, le calcul de $\sigma(T_i)$, pouvant être répétée après l'enlèvement d'une trace malveillant de manière à vérifier si une autre peut aussi être détectée.

Appendix B

PePiT: code snippet and UML class diagram

Listing B.1: PePiT: how to start/join an ad hoc network for Galaxy S II devices.

```
1  private ShellInterface shell = new ShellInterface();
2  // ...
3  switch (phoneType) {
4          // ...
5          case SAMSUNG_GALAXY_S_II:
6                  shell.startCommand("insmod /lib/modules/dhd.ko \"firmware_path=/↩
                       system/etc/wifi/bcm4330_sta.bin nvram_path=/system/etc/wifi/↩
                       nvram_net.txt\"");
7                  shell.startCommand(APP_PATH+"/bin/iwconfig eth0 essid " + ↩
                       ESSID_NAME_S + " mode ad-hoc");
8                  shell.startCommand(APP_PATH+"/bin/iwconfig eth0 channel 1");
9                  shell.startCommand(APP_PATH+"/bin/iwconfig eth0 commit");
10                 shell.startCommand(APP_PATH+"/bin/ifconfig eth0 " + this.mAddress↩
                        + " netmask 255.255.0.0");
11                 break;
```

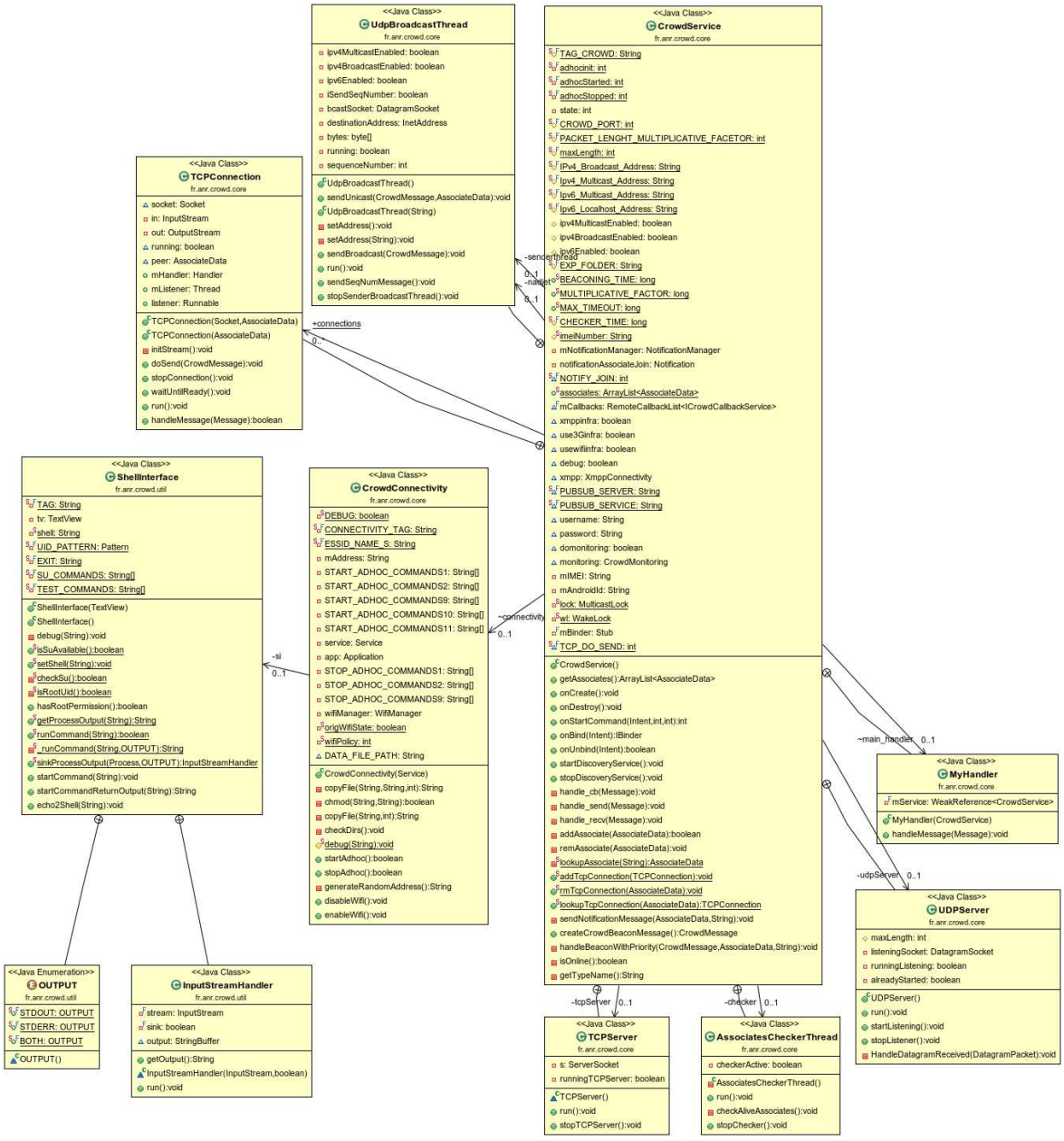


Figure B.1: PePiT communication module UML class diagram.

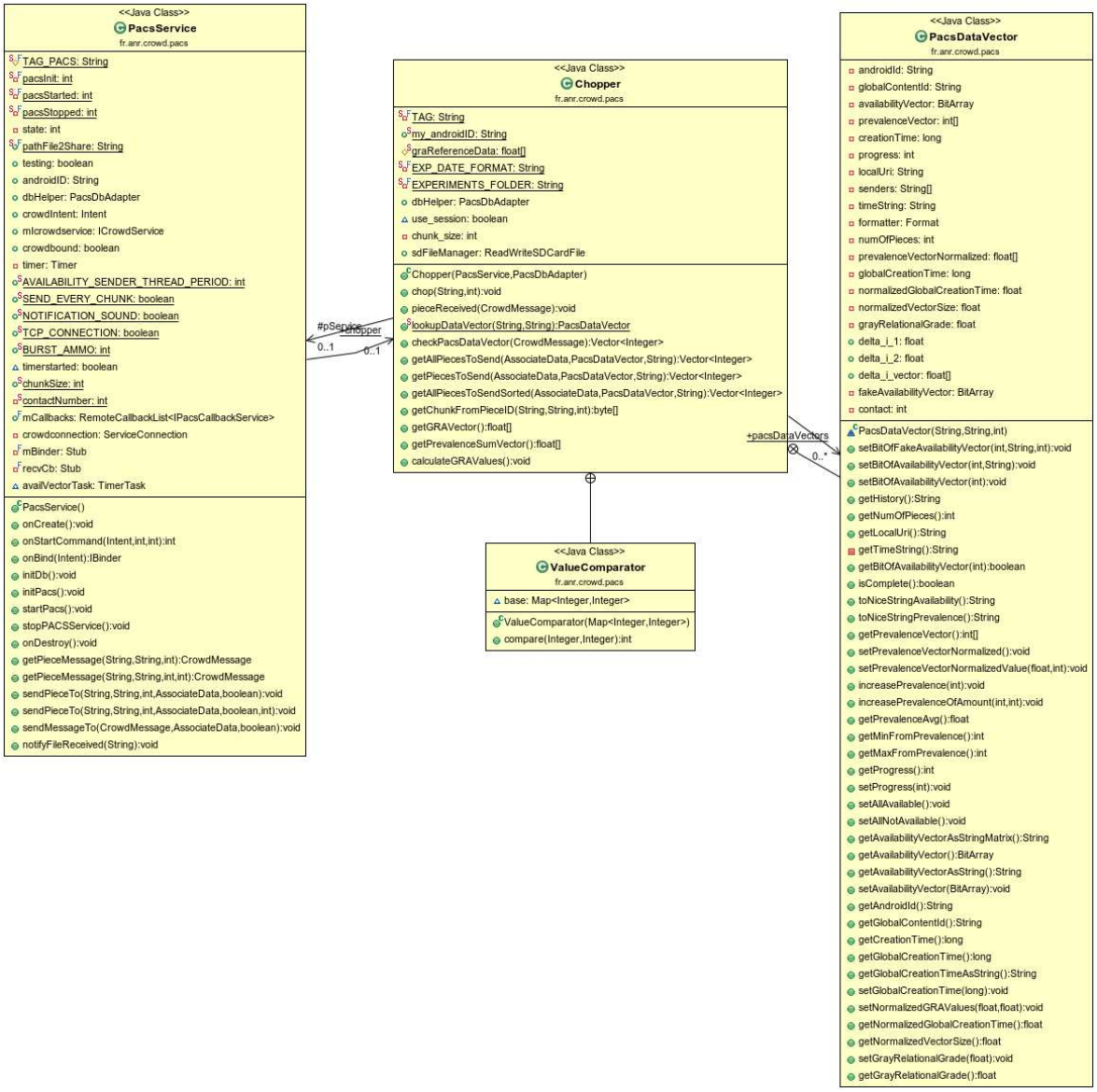


Figure B.2: PePiT communication module UML class diagram.

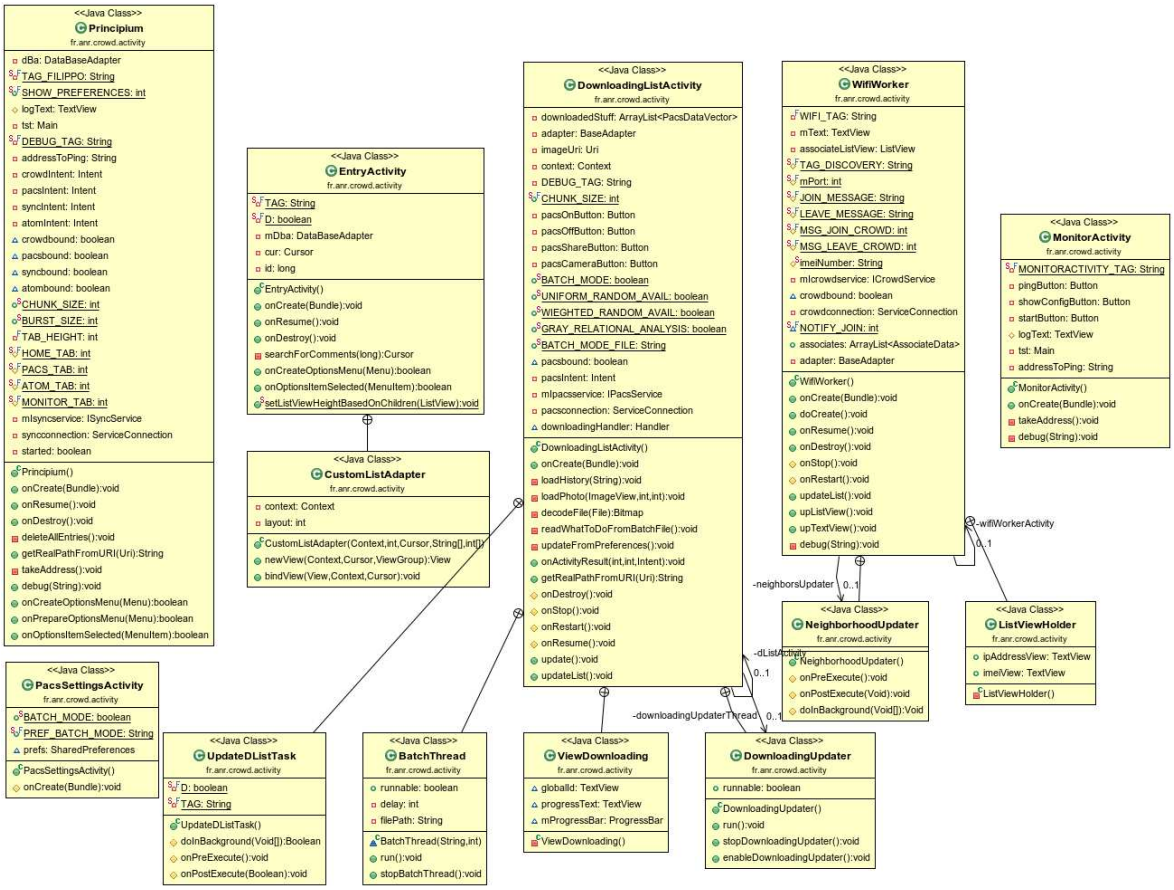


Figure B.3: PePiT communication module UML class diagram.

Appendix C

List of publications

C.1. Published

Matteo Sammarco, Miguel Elias Mitre Campista and Marcelo Dias de Amorim, *Trace Selection for Improved WLAN Monitoring* in ACM Sigcomm 2013, HotPlanet Workshop, Hong Kong, China, August 2013.

Matteo Sammarco, Miguel Elias Mitre Campista and Marcelo Dias de Amorim, *Towards a Scalable WiFi Monitoring System*, in WNetVirt 2013, Angra dos Reis, Rio de Janeiro, Brazil, October 2013.

Matteo Sammarco, Nadjet Belblidia, Yoann Lopez, Marcelo Dias de Amorim, Luís Henrique Maciel Kosmalski Costa, and Jérémie Leguay, *PePiT: Opportunistic Dissemination of Large Contents on Android Mobile Devices*, ACM MobiOpp workshop Demo Session, Zurich, Switzerland, March 2012.

Matteo Sammarco, Miguel Elias Mitre Campista and Marcelo Dias de Amorim, *Highly Scalable Wi-Fi Monitoring Systems*, Ecole d'été ResCom, Porquerolles, France, May 2013.

C.2. Under review

Miguel Elias Mitre Campista, **Matteo Sammarco**, Marcelo Dias de Amorim and Tahiry Razafindralambo, *Collaborative Wireless Measurements at Risk: Vulnerabilities When Users Come at Play*, under review (3rd round) at IEEE Transactions on Parallel and Distributed Systems, July 2013.

Matteo Sammarco, Miguel Elias Mitre Campista and Marcelo Dias de Amorim, *Scalable Wireless Traffic Capture Through Community Detection and Trace Similarity*, under review at IEEE Transactions on Mobile Computing, October 2013.

Nadjet Belblidia, **Matteo Sammarco**, Luís Henrique Maciel Kosmalski Costa and Marcelo Dias de Amorim, *Opportunistic Multi-Content Dissemination*, under review at IEEE Transactions on Mobile Computing, October 2013.

Miguel Elias Mitre Campista, **Matteo Sammarco**, Marcelo Dias de Amorim and Tahiry Razafindralambo, *Monitoramento Colaborativo de Redes Sem-fio: Acurácia do Sistema e Denúncia de Farejadores Maliciosos*, under review at the 32nd Brazilian Symposium on Computer Networks and Distributed Systems.

Nadjet Belblidia, **Matteo Sammarco**, Luís Henrique Maciel Kosmalski Costa and Marcelo Dias de Amorim, *Disseminação Oportunista de Múltiplos Conteúdos de Grande Porte Usando a Análise Relacional Cinza*, under review at the 32nd Brazilian Symposium on Computer Networks and Distributed Systems.

C.3. Other publications

Dario Rossi, Paolo Veglia, **Matteo Sammarco** and Federico Larroca, *ModelNet-TE: An emulation tool for the study of P2P and Traffic Engineering interaction dynamics*. Springer Peer-to-peer Networking and Applications (PPNA), June 2013, Volume 6, Issue 2, pp 194-212.

Bibliography

[1] URL `http://www.android-x86.org`.

[2] URL `http://www.podnet.ee.ethz.ch/scope/start`.

[3] Wireless lan equipment and wifi phones quartely market share, size, and forecasts. White Paper.

[4] Cisco visual networking index: Global mobile data traffic forecast update, 2012–2017. White Paper, 2013.

[5] Prashanth A. K. Acharya, Ashish Sharma, Elizabeth M. Belding, Kevin C. Almeroth, Senior Member, and Dina Papagiannaki. Rate adaptation in congested wireless networks through real-time measurements. *IEEE Transactions on Mobile Computing*, 9(11): 1535–1550, November 2010.

[6] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *SOCIAL NETWORKS*, 25:211–230, 2001.

[7] Adnan Agbaria and Roy Friedman. Efficient and reliable dissemination in wireless opportunistic networks by location extrapolation. In *ACM MobiOpp*, pages 101–109, Pisa, Italy, 2010.

[8] Panayotis Antoniadis, Bénédicte Le Grand, Anna Satsiou, Leandros Tassiulas, Rui Aguiar, Joã Paulo Barraca, and S. Sargento. Community building over neighborhood wireless mesh networks. *IEEE Technology and Society*, 27(1):48–56, February 2008.

[9] A. Asterjadhi, E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. Toward network coding-based protocols for data broadcasting in wireless Ad Hoc networks. *IEEE Transactions on Wireless Communications*, 9(2), Feb. 2010.

[10] Anand Balachandran, Geoffrey M. Voelker, Paramvir Bahl, and P. Venkat Rangan. Characterizing user behavior and network performance in a public wireless LAN. In *ACM SIGMETRICS*, pages 195–205, June15-19 2002.

[11] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. DTN routing as a resource allocation problem. In *ACM Sigcomm*, pages 373–384, Kyoto, Japan, 2007.

[12] M Bastian, S Heymann, and M Jacomy. Gephi: an open source software for exploring and manipulating networks. 2009. In *International AAAI Conference on Weblogs and Social Media*, 2011.

[13] Nadjet Belblidia. *Capacity-Aware Opportunistic Networks: Characterization and Impact on the Dissemination of Large Contents*. PhD thesis, University Pierre and Marie Curie, 2012.

[14] Nadjet Belblidia, Marcelo Dias de Amorim, Luís H. M. K. Costa, Jeremie Leguay, and Vania Conan. PACS: Chopping and shuffling large contents for faster opportunistic dissemination. In *IFIP/IEEE Annual Conference on Wireless On-demand Network Systems and Services*, pages 9–16, Bardonecchia, Italy, 2011.

[15] Nadjet Belblidia, Marcelo Dias de Amorim, Luís H. M. K. Costa, Jeremie Leguay, and Vania Conan. Part-whole dissemination of large multimedia contents in opportunistic networks. *Computer Communications*, Mar. 2012.

[16] C. Boldrini, M. Conti, F. Delmastro, and A. Passarella. Context- and social-aware middleware for opportunistic networks. *J. Netw. Comput. Appl.*, 33(5):525–541, September 2010. ISSN 1084-8045. doi: 10.1016/j.jnca.2010.03.017. URL `http://dx.doi.org/10.1016/j.jnca.2010.03.017`.

[17] Chiara Boldrini, Marco Conti, and Andrea Passarella. Contentplace: social-aware data dissemination in opportunistic networks. In *ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 203–210, Vancouver, Canada, 2008.

[18] Vincent Borrel, Franck Legendre, Marcelo Dias de Amorim, and Serge Fdida. Simps: using sociology for personal mobility. *IEEE/ACM Trans. Netw.*, 17(3):831–842, 2009. URL `http://dblp.uni-trier.de/db/journals/ton/ton17.html#BorrelLAF09`.

[19] John Burgess, Brian Gallagher, David Jensen, and Brian N. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking. In *IEEE Infocom*, pages 1–11, Barcelona, Spain, 2006.

[20] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6):128–136, Jun. 2003.

[21] Miguel Elias M. Campista, Marcelo Dias de Amorim, and Luís Henrique M. K. Costa. Big wireless measurement campaigns: Are they really worth the price? In *ACM International Workshop on Hot Topics in Planet-Scale Measurements (ACM HotPlanet)*, pages 27–32, June25 2012.

[22] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. RFC 4838 (Informational), April 2007. URL `http://www.ietf.org/rfc/rfc4838.txt`.

[23] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 1–14, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-908-1. doi: 10.1145/1298306.1298309. URL `http://doi.acm.org/10.1145/1298306.1298309`.

[24] Yu-Chung Cheng, John Bellardo, Péter Benkö, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *ACM SIGCOMM*, pages 39–50, September11-15 2006.

[25] Thomas Claveirole and Marcelo Dias de Amorim. WiPal: IEEE 802.11 traces manipulation software, January 2010. URL `http://wipal.lip6.fr/`.

[26] Thomas Claveirole and Marcelo Dias de Amorim. Manipulating Wi-Fi packet traces with WiPal: design and experience. *Software Practice & Experience*, 42(5):585–599, May 2012.

[27] William Cook. Concorde tsp solver. *See: http://www. tsp. gatech. edu/concorde. html*, 2005.

[28] M. Cunche, M.-A. Kaafar, and R. Boreli. I know who you will meet this evening! linking wireless devices using wi-fi probe requests. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–9, June. doi: 10.1109/WoWMoM.2012.6263700.

[29] D. A. Darling. The Kolmogorov-Smirnov, Cramér-von Mises Tests. *The Annals of Mathematical Statistics*, 28(4):823–838, December 1957.

[30] R. B. Dean and W. J. Dixon. Simplified statistics for small numbers of observations. *Analytical Chemistry*, 23(4):636–638, April 1951.

[31] J. L. Deng. Control problems of grey systems. *Systems & Control Letters*, 1(5):288–294, 1982.

[32] J. L. Deng. Introduction to grey system theory. *Journal of Grey System*, 1:1–24, Nov. 1989.

[33] Christina Fragouli, Jörg Widmer, and Jean-Yves Le Boudec. Efficient broadcasting using network coding. *IEEE/ACM Transactions on Networking*, 16(2):450–463, 2008.

[34] N. Gaddam and A. Potluri. Study of BitTorrent for file sharing in Ad Hoc networks. In *IEEE Wireless Communications and Networking Conference*, pages 1–6, Allahabad, India, 2009.

[35] Adriano Galati and Chris Greenhalgh. CRAWDAD data set nottingham/mall (v. 2013-02-05). Downloaded from http://crawdad.cs.dartmouth.edu/nottingham/mall, February 2013.

[36] S.K. Goel, M. Singh, Dongyan Xu, and Baochun Li. Efficient peer-to-peer data dissemination in mobile Ad Hoc networks. In *International Conference on Parallel Processing Workshops*, pages 152–158, Vancouver, Canada, 2002.

[37] graphviz. Graphviz - Graph Visualization Software, September 2012. URL `http://www.graphviz.org`.

[38] Nidhi Hegde, Fabien Mathieu, and Diego Perino. Size does matter (in p2p live streaming). *CoRR*, abs/0909.1713, 2009.

[39] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *ACM Mobicom*, pages 174–185, Seattle, Washington, United States, 1999.

[40] Ólafur R. Helgason, Emre A. Yavuz, Sylvia T. Kouyoumdjieva, Ljubica Pajevic, and Gunnar Karlsson. A mobile peer-to-peer system for opportunistic content-centric networking. In *ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*, pages 21–26, New Delhi, India, 2010.

[41] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. In *ACM MobiCom*, pages 187–201, Sep.-Oct.26-1 2004.

[42] Camden C. Ho, Krishna N. Ramachandran, Kevin C. Almeroth, and Elizabeth M. Belding-Royer. A scalable framework for wireless network monitoring. In *Proceedings of the 2nd ACM international workshop on Wireless mobile applications and services on WLAN hotspots*, WMASH '04, pages 93–101, New York, NY, USA, 2004. ACM. ISBN 1-58113-877-6. doi: 10.1145/1024733.1024745. URL `http://doi.acm.org/10.1145/1024733.1024745`.

[43] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket switched networks and human mobility in conference environments. In *ACM SIGCOMM workshop on Delay-tolerant networking*, pages 244–251, Philadelphia, Pennsylvania, USA, 2005.

[44] Pan Hui, Augustin Chaintreau, Richard Gass, James Scott, Jon Crowcroft, and Christophe Diot. Pocket switched networking: Challenges, feasibility and implementation issues. In *Proceedings of the Second International IFIP Conference on Autonomic Communication*, WAC'05, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-32992-7, 978-3-540-32992-3. doi: 10.1007/11687818_1. URL `http://dx.doi.org/10.1007/11687818_1`.

[45] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *ACM Mobihoc*, pages 241–250, Hong Kong, China, 2008.

[46] E. Hyytia, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott. When does content float? characterizing availability of anchored information in opportunistic content sharing. In *IEEE Infocom*, pages 3137–3145, Shanghai, China, 2011.

[47] S. Ioannidis, A. Chaintreau, and L. Massoulie. Optimal and scalable distribution of content updates over a mobile social network. In *IEEE Infocom*, pages 1422–1430, Rio de Janeiro, 2009.

[48] Stratis Ioannidis, Laurent Massoulie, and Augustin Chaintreau. Distributed caching over heterogeneous mobile networks. In *ACM Sigmetrics*, pages 311–322, New York, NY, USA, 2010.

[49] Behrouz Jedari and Feng Xia. A survey on routing and data dissemination in opportunistic mobile social networks. *CoRR*, abs/1311.0347, 2013.

[50] Sewook Jung, Uichin Lee, Alexander Chang, Dae-Ki Cho, and Mario Gerla. Bluetorrent: Cooperative content sharing for bluetooth users. In *IEEE International Conference on Pervasive Computing and Communications*, pages 47–56, New York, USA, 2007.

[51] Aman Kansal, Michel Goraczko, and Feng Zhao. Building a sensor network of mobile phones. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, IPSN '07, pages 547–548, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-638-7. doi: 10.1145/1236360.1236433. URL `http://doi.acm.org/10.1145/1236360.1236433`.

[52] Gunnar Karlsson, Vincent Lenders, and Martin May. Delay-tolerant broadcasting. *IEEE Transactions on Broadcasting*, 53(1):369–381, Mar. 2007.

[53] Abdelmajid Khelil, Christian Becker, Jing Tian, and Kurt Rothermel. An epidemic model for information diffusion in manets. In *ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 54–60, Atlanta, Georgia, USA, 2002.

[54] A. Klemm, C. Lindemann, and O.P. Waldhorst. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In *IEEE Vehicular Technology Conference*, volume 4, pages 2758–2763, Orlando, USA, 2003.

[55] Vinod Kone, Mariya Zheleva, Mile Wittie, Ben Y. Zhao, Elizabeth M. Belding, Haitao Zheng, and Kevin Almeroth. AirLab: consistency, fidelity and privacy in wireless measurements. *SIGCOMM Comput. Commun. Rev.*, 41(1):60–65, January 2011.

[56] Iordanis Koutsopoulos. Optimal incentive-driven design of participatory sensing systems. In *INFOCOM*, pages 1402–1410, 2013.

[57] Amir Krifa, Chadi Barakat, and Thrasyvoulos Spyropoulos. Optimal Buffer Management Policies for Delay Tolerant Networks. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, San Francisco, USA, 2008.

[58] Amir Krifa, Mohamed Karim Sbai, Chadi Barakat, and Thierry Turletti. Bithoc: A content sharing application for wireless ad hoc networks. In *IEEE International Conference on Pervasive Computing and Communications*, pages 1–3, Galveston, TX, USA, 2009.

[59] Nikolaos Laoutaris, Damiano Carra, and Pietro Michiardi. Uplink allocation beyond choke/unchoke or how to divide and conquer best, 2008.

[60] K.C. Lee, Seung-Hoon Lee, Ryan Cheung, Uichin Lee, and M. Gerla. First experience with cartorrent in a real vehicular ad hoc network testbed. In *IEEE Mobile Networking for Vehicular Environments*, pages 109–114, Anchorage, Alaska, USA, 2007.

[61] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. Code torrent: content distribution using network coding in vanet. In *International workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5, Los Angeles, California, 2006.

[62] Uichin Lee, Sewook Jung, Dae-Ki Cho, A. Chang, Junho Choi, and M. Gerla. P2P content distribution to mobile bluetooth users. *IEEE Transactions on Vehicular Technology*, 59(1):356–367, Jan. 2010.

[63] Jérémie Leguay, Timur Friedman, and Vania Conan. Dtn routing in a mobility pattern space. In *ACM SIGCOMM workshop on Delay-tolerant networking*, pages 276–283, Philadelphia, Pennsylvania, USA, 2005.

[64] Vincent Lenders, Martin May, Gunnar Karlsson, and Clemens Wacha. Wireless ad hoc podcasting. *ACM Mobile Computing and Communications Review*, 12:65–67, Jan. 2008.

[65] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7:19–20, Jul. 2003.

[66] Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan. Analyzing the MAC-level behavior of wireless networks in the wild. In *ACM SIGCOMM*, pages 75–86, September11-15 2006.

[67] Pawel Marciniak, Nikitas Liogkas, Arnaud Legout, and Eddie Kohler. Small is not always beautiful. *CoRR*, abs/0802.1015, 2008.

[68] Abraham Martín-Campillo, Jon Crowcroft, Eiko Yoneki, Ramon Martí, and Carlos Martínez-García. Using haggle to create an electronic triage tag. In *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp '10, pages 167–170, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-925-1. doi: 10.1145/1755743.1755775. URL `http://doi.acm.org/10.1145/1755743.1755775`.

[69] Syed Haani Masood, Syed Ali Raza, and Mark Coates. Content distribution strategies in opportunistic networks. *CoRR*, abs/1308.0786, 2013.

[70] Paolo Meroni, Elena Pagani, Gian Paolo Rossi, and Lorenzo Valerio. An opportunistic platform for android-based mobile devices. In *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp '10, pages 191–193, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-925-1. doi: 10.1145/1755743.1755783. URL http://doi.acm.org/10.1145/1755743.1755783.

[71] Arezu Moghadam, Suman Srinivasan, and Henning Schulzrinne. 7ds - a modular platform to develop mobile disruption-tolerant applications. In *Proceedings of the 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, NGMAST '08, pages 177–183, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3333-9. doi: 10.1109/NGMAST.2008.75. URL http://dx.doi.org/10.1109/NGMAST.2008.75.

[72] Mirco Musolesi and Cecilia Mascolo. Controlled Epidemic-Style Dissemination Middleware for Mobile Ad Hoc Networks. In *International Conference on Mobile and Ubiquitous Systems: Networks and Services*, pages 1–9, San Jose, CA, USA, 2006.

[73] A. Nandan, S. Das, G. Pau, M. Gerla, and M.Y. Sanadidi. Co-operative downloading in vehicular Ad Hoc wireless networks. In *IFIP/IEEE Annual Conference on Wireless On-demand Network Systems and Services*, pages 32–41, St.Moritz, Switzerland, 2005.

[74] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113+, February 2004. doi: 10.1103/physreve.69.026113. URL http://dx.doi.org/10.1103/physreve.69.026113.

[75] Jörg Ott and Jussi Kangasharju. Opportunistic content sharing applications. In *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, pages 19–24, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1291-2. doi: 10.1145/2248361.2248367. URL http://doi.acm.org/10.1145/2248361.2248367.

[76] Charith Perera, Arkady B. Zaslavsky, Peter Christen, Ali Salehi, and Dimitrios Georgakopoulos. Capturing sensor data from mobile phones using global sensor network middleware. In *PIMRC*, pages 24–29. IEEE, 2012. ISBN 978-1-4673-2566-0. URL http://dblp.uni-trier.de/db/conf/pimrc/pimrc2012.html#PereraZCSG12.

[77] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, and Christophe Diot. Mobiclique: Middleware for mobile social networking. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, pages 49–54, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-445-4. doi: 10.1145/1592665.1592678. URL http://doi.acm.org/10.1145/1592665.1592678.

[78] Mikko Pitkänen, Ari Keränen, and Jörg Ott. Message fragmentation in opportunistic dtns. In *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, pages 1–7, Newport Beach, CA, USA, 2008.

[79] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *Journal of Graph Algortihms and Applications*, 10(2):191–218, 2006. doi: 10.7155/jgaa.00124.

[80] Valentin Radu, Lito Kriara, and Mahesh K. Marina. Pazl: A mobile crowdsensing based indoor wifi monitoring system. In *CNSM*, pages 75–83, 2013.

[81] Usha N. Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106+, September 2007. doi: 10.1103/physreve.76.036106. URL `http://dx.doi.org/10.1103/physreve.76.036106`.

[82] R. Raghavendra, J. Padhye, R. Mahajan, and E. Belding. Wi-Fi Networks are Underutilized. Technical report, Microsoft Research, 2009.

[83] Sundaram Rajagpalan and Chien Chung Shen. A cross-layer decentralized BitTorrent for mobile Ad Hoc networks. In *International Conference on Mobile and Ubiquitous Systems: Networks and Services*, pages 203–212, San Jose, USA, 2006.

[84] V. Ramiro, E. Lochin, P. Senac, and T. Rakotoarivelo. On the limits of dtn monitoring. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–6, 2013. doi: 10.1109/WoWMoM.2013.6583441.

[85] Juan Ramos. Using TF-IDF to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, 2003.

[86] Tahiry Razafindralambo, Thomas Begin, Marcelo Dias de Amorim, Isabelle Guérin Lassous, Nathalie Mitton, and David Simplot-Ryl. Promoting quality of service in substitution networks with controlled mobility. In *Ad Hoc Networks and Wireless (ADHOC-NOW)*, pages 248–261, July18–20 2011.

[87] Joshua Reich and Augustin Chaintreau. The age of impatience: optimal replication schemes for opportunistic networks. In *ACM Conext*, pages 85–96, Rome, Italy, 2009.

[88] Marco Di Renzo, Harald Haas, and Peter M. Grant. Spatial modulation for multiple-antenna wireless systems: a survey. *IEEE Communications Magazine*, 49(12):182–191, 2011. URL `http://dblp.uni-trier.de/db/journals/cm/cm49.html#RenzoHG11`.

[89] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seongjoon Kim, and Song Chong. CRAWDAD trace ncsu/mobilitymodels/gps/kaist (v. 2009-07-23). Downloaded from http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels/GPS/KAIST, July 2009.

[90] Nikodin Ristanovic, George Theodorakopoulos, and Jean-Yves Le Boudec. Traps and pitfalls of using contact traces in performance studies of opportunistic networks. In *INFOCOM*, pages 1377–1385, 2012.

[91] Joshua Robinson, Ram Swaminathan, and Edward W. Knightly. Assessment of urban-scale wireless networks with a small number of measurements. In *ACM MobiCom*, pages 187–198, September14–19 2008.

[92] Bilel Ben Romdhanne, Diego Dujovne, and Thierry Turletti. Efficient and scalable merging algorithms for wireless traces. In *Workshop on Real Overlays and Distributed Systems (ROADS)*, pages 1–7, October 2009.

[93] Daniel J. Rosenkrantz, Richard Edwin Stearns, and Philip M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, pages 563–581, 1977.

[94] M Rosvall and C T Bergstrom. Maps of information flow reveal community structure in complex networks. Technical Report arXiv:0707.0609, Jul 2007. Comments: 6 pages and 3 figures.

[95] Matteo Sammarco, Nadjet Belblidia, Yoann Lopez, Marcelo Dias de Amorim, Luís H. M. K. Costa, and Jeremie Leguay. PePiT: Opportunistic Dissemination of Large Contents on Android Mobile Devices. In *ACM MobiOpp Demo Session*, Zurich, Switzerland, 2012.

[96] Matteo Sammarco, Miguel E. M. Campista, and Marcelo Dias de Amorim. Trace selection for improved WLAN monitoring. In *Proceedings of the 5th ACM workshop on HotPlanet*, HotPlanet '13, pages 9–14, New York, New York, USA, 2013. ACM Press. ISBN 9781450321778. doi: 10.1145/2491159.2491165. URL `http://dx.doi.org/10.1145/2491159.2491165`.

[97] Mohamed Sbai, Emna Salhi, and Chadi Barakat. P2P content sharing in spontaneous multi-hop wireless networks. In *International conference on Communication systems and Networks*, pages 1–10, Bangalore, India, 2010.

[98] Aaron Schulman, Dave Levin, and Neil Spring. On the fidelity of 802.11 packets traces. In *International conference on Passive and Active network Measurement (PAM)*, pages 132–141, April 2008.

[99] P. Serrano, M. Zink, and J. Kurose. Assessing the Fidelity of COTS 802.11 Sniffers. In *INFOCOM 2009, IEEE*, pages 1089–1097, April. doi: 10.1109/INFCOM.2009.5062021.

[100] Yong Sheng, Guanling Chen, Hongda Yin, K. Tan, U. Deshpande, B. Vance, D. Kotz, A. Campbell, C. McDonald, T. Henderson, and J. Wright. MAP: a scalable monitoring system for dependable 802.11 wireless networks. *Wireless Communications, IEEE*, 15 (5):10–18, October 2008. ISSN 1536-1284. doi: 10.1109/MWC.2008.4653127.

[101] IEEE Computer Society. IEEE standard 802.11 part 11: Wireless LAN Medium Access Control (MAC) and Phisical Layer (PHY) Specifications, 2012.

[102] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating location predictors with extensive wi-fi mobility data. *Mobile Computing and Communications Review*, 7(4):64–65, 2003. URL `http://dblp.uni-trier.de/db/journals/sigmobile/sigmobile7.html#SongKJH03`.

[103] Hasan Sözer, Metin Tekkalmaz, and Ibrahim Korpeoglu. A peer-to-peer file search and download protocol for wireless ad-hoc networks. *Computer Communications*, 32: 41–50, Jan. 2009.

[104] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi Raghavendra. Single-copy routing in intermittently connected mobile networks. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 235–244, Santa Clara, CA, USA, 2004.

[105] Kannan Srinivasan, Maria A. Kazandjieva, Mayank Jain, Edward Kim, and Philip Levis. SWAT: enabling wireless network measurements. In *ACM conference on Embedded network sensor systems (SenSys)*, pages 395–396, November5–7 2008.

[106] Jing Su, James Scott, Pan Hui, Jon Crowcroft, Eyal De Lara, Christophe Diot, Ashvin Goel, Meng How Lim, and Eben Upton. Haggle: Seamless networking for mobile applications. In *Proceedings of the 9th International Conference on Ubiquitous Computing*, UbiComp '07, pages 391–408, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-74852-6. URL `http://dl.acm.org/citation.cfm?id=1771592.1771615`.

[107] Keren Tan, Chris McDonald, Bennet Vance, Chrisil Arackaparambil, Sergey Bratus, and David Kotz. From MAP to DIST: the evolution of a large-scale WLAN monitoring system. *IEEE Transactions on Mobile Computing*, 99(PrePrints):1, 2012. ISSN 1536-1233. doi: http://doi.ieeecomputersociety.org/10.1109/TMC.2012.237.

[108] TaskBomb v1.8.12. Taskbomb v1.8.12. `http://androidideas.org/taskbomb`, 2012.

[109] Pierre Ugo Tournoux, Jérémie Leguay, Farid Benbadis, Vania Conan, Marcelo Dias de Amorim, and John Whitbeck. The accordion phenomenon: Analysis, characterization, and impact on DTN routing. In *IEEE Infocom*, pages 1116–1124, Rio de Janeiro, 2009.

[110] Sacha Trifunovic, Bernhard Distl, Dominik Schatzmann, and Franck Legendre. Wifi-opp: ad-hoc-less opportunistic networking. In *ACM CHANTS*, pages 37–42, Las Vegas, Nevada, USA, 2011.

[111] Cristian Tuduce and Thomas R. Gross. A mobility model based on wlan traces and its validation. In *INFOCOM*, pages 664–674. IEEE, 2005. URL `http://dblp.uni-trier.de/db/conf/infocom/infocom2005.html#TuduceG05`.

[112] Bryan Veal, Kang Li, and David K. Lowenthal. New methods for passive estimation of tcp round-trip times. In Constantinos Dovrolis, editor, *PAM*, volume 3431 of *Lecture Notes in Computer Science*, pages 121–134. Springer, 2005. ISBN 3-540-25520-6. URL `http://dblp.uni-trier.de/db/conf/pam/pam2005.html#VealLL05`.

[113] Pedro B. Velloso, Rafael P. Laufer, Daniel de Oliveira Cunha, Otto Carlos M. B. Duarte, and Guy Pujolle. Trust management in mobile ad hoc networks using a scalable maturity-based model. *IEEE Transactions on Network and Service Management*, 7(3): 172–185, 2010.

[114] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *ACM Mobihoc*, pages 194–205, Lausanne, Switzerland, 2002.

[115] Bo Xing, Karim Seada, and Nalini Venkatasubramanian. Proximiter: Enabling mobile proximity-based content sharing on portable devices. In *PerCom*, pages 1–3. IEEE Computer Society, 2009. ISBN 978-1-4244-3304-9. URL `http://dblp.uni-trier.de/db/conf/percom/percom2009.html#XingSV09`.

[116] Jihwang Yeo, Moustafa Youssef, and Ashok Agrawala. A framework for wireless lan monitoring and its applications. In *Proceedings of the 3rd ACM Workshop on Wireless Security*, WiSe '04, pages 70–79, New York, NY, USA, 2004. ACM. ISBN 1-58113-925-X. doi: 10.1145/1023646.1023660. URL `http://doi.acm.org/10.1145/1023646.1023660`.

[117] Chih-Wei Yi. A unified analytic framework based on minimum scan statistics for wireless ad hoc and sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(9):1233–1245, September 2009.

[118] Eiko Yoneki, Pan Hui, ShuYan Chan, and Jon Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 225–234, Chania, Crete Island, Greece, 2007.

[119] Xiaolan Zhang, Jim Kurose, Brian Neil Levine, Don Towsley, and Honggang Zhang. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In *ACM Mobicom*, pages 195–206, Montré;al, Québec, Canada, 2007.

[120] Zhensheng Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *IEEE Communications Surveys and Tutorials*, 8(1):24–37, quarter 2006.

[121] J. Zhao and G. Cao. Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks. In *IEEE Infocom*, pages 1–12, 2006.

[122] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *ACM Mobihoc*, pages 187–198, Roppongi Hills, Tokyo, Japan, 2004.

[123] Ruogu Zhou, Guoliang Xing, Xunteng Xu, Jianping Wang, and Lin Gu. Wiznet: A zigbee-based sensor system for distributed wireless lan performance monitoring. *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 0:123–131, 2013. doi: http://doi.ieeecomputersociety.org/10.1109/PerCom.2013.6526722.

[124] Gjergji Zyba, Geoffrey M. Voelker, Stratis Ioannidis, and Christophe Diot. Dissemination in Opportunistic Mobile Ad-hoc Networks: the Power of Crowd. In *IEEE Infocom*, pages 1179–1187, Shanghai, China, 2011.

List of Figures

1.1. Problems occurring in wireless communications. 2
1.2. Wireless traffic capturing during opportunistic content exchange 4
1.3. Experimental workflow. 6

3.1. Modules of the PePiT architecture. 20
3.2. Crowd message format. 22
3.3. PePiT user interface. 23
3.4. An illustrative scenario of the working of EPICS. 24
3.5. Distribution of completion times for experiments. 27
3.6. Content dissemination delays. 28
3.7. Contacts and inter-contacts in the emulated mobile scenario. All nodes start at the same time. 29
3.8. EPICS versus Uniform in an emulated mobile scenario. Contents have variable size and variable creation times. 30
3.9. Mobile scenario plan. Path is highlighted in red. All nodes start at the same time. 31
3.10. EPICS versus Uniform in a mobile scenario. Contents have variable size and variable creation times. 31

4.1. Screenshot of PePiT option settings. 34
4.2. Dissemination time by tuning the piece size and using UDP sockets. 36
4.3. Data-link and application level measurements tuning the piece size and using UDP sockets. 37
4.4. Dissemination time tuning the piece size and using TCP sockets. 38
4.5. Average dissemination difference using TCP and UDP sockets. 39
4.6. Dissemination efficiency tuning the piece size and using TCP sockets. 40
4.7. Dissemination time and external traffic with burst mode activated. 40
4.8. View of the transmission queue. With a burst size = 10, pieces experience a long queue delay and they result useless once transmitted. 40
4.9. Dissemination time of one content, tuning the burst size and changing the number of nodes involved. 41
4.10. Operation area: Burst size for minimum diffusion time with a tolerance of 30 seconds. 42
4.11. Cumulative distribution function of the number of nodes that every node perceives in his neighborhood, including itself. (a) Shopping Mall trace, (b) KAIST trace, (c) SIMPS trace. 42
4.12. SIMPS simulator. 43
4.13. DAD vs. EPICS and vs. EPICS with a burst of ten pieces. CDF of dissemination times per content and per node. 44

5.1. RTS-CTS mechanism and fragmentation. 51

5.2. MAC frame format and frame control field structure. 51
5.3. Typical wireless measurement system architecture: sensing, merge, and presentation modules. 52
5.4. Time elapsed after a complete merging procedure for an increasing number of traces. Average time and 95% confidence interval are calculated over 10 experiments. Reducing the number of traces can significantly affect the time required for the entire operation. 53

6.1. Impact of merging a selected subset of traces. In each trace, the collected and missed frames are represented by black and white rectangles, respectively. Merging all traces together permits more complete results but also incurs in more merging operations. 58
6.2. Additional tasks of the proposed merging procedure. 59
6.3. Monitor deployment for IRCICA and INRIA scenarios. 62
6.4. Wireless traffic characterization in IRCICA and INRIA scenarios. 63
6.5. IRCICA scenario, similarity matrices. (a) Intra-flow, (b) Inter-flow, (c) Adamic, (d) Power, (e) Weighted inter-flow. 65
6.6. INRIA scenario, similarity matrices. (a) Intra-flow, (b) Inter-flow, (c) Adamic, (d) Power, (e) Weighted inter-flow. 66
6.7. Graphs generated using traces as nodes and weighted inter-flow similarity values as edge lengths. Different node shapes (◆, ●, ■) denote different communities. 71
6.8. Comparison of the merging process performance ranking traces according to the size and the TSP solution. 73
6.9. A fully connected graph with 5 nodes belonging to 2 communities. Higher the similarity between traces, thicker the edges connecting them. 74
6.10. Comparison of the merging process performances ranking traces according to the ascending node degree and the TSP solution. 76

7.1. The fraction of captured frames increases after merging two individual traces from different sensing nodes. Each individual sensing node, S_1 and S_2, has captured 50% of the total frames numbered in sequential order from 1 to 4. After merging, the fraction of captured frames increases to 75%. 81
7.2. Repulsive attack. Malicious node M forges a trace containing a complete sequence of frames. As a consequence, the fraction of captured frames increases from 75% to 83% and the wireless infrastructure is repulsed to other areas. At the end, the wireless infrastructure becomes farther away from M. 82
7.3. Attractive attack. Malicious node M forges a trace containing an empty sequence of frames. As a consequence, the fraction of captured frames reduces from 75% to 66% and the wireless infrastructure is attracted toward M. At the end, the wireless infrastructure becomes closer to M. 82
7.4. Experimental scenario. 85
7.5. Fraction of received frames considering all source-destination pairs in both scenarios. 88
7.6. Sequence number variation considering all source-destination pairs in both scenarios. 88
7.7. Geographical distribution of percentage of received frames considering all source-destination pairs in both scenarios. 89
7.8. Geographical distribution of percentage of received frames for sorted fraction of received frames considering all source-destination pairs in both scenarios. 89

7.9. Impact of inserting traces with fake communicating pairs on the accuracy of the merged trace. . . . 90
7.10. Node strength variation in repulsive attack. . . . 91
7.11. Node strength variation in attractive attack. . . . 91
7.12. Graph obtained considering a malicious node executing the repulsive attack in the collocated scenario: (a) with 25 fake communicating pairs and (b) with 400 fake communicating pairs. . . . 92
7.13. Graph obtained considering a malicious node executing the repulsive attack in the scattered scenario: (a) with 25 fake communicating pairs and (b) with 400 fake communicating pairs. . . . 93
7.14. Graph obtained considering a malicious node executing the attractive attack in the collocated scenario: (a) with 25 fake communicating pairs and (b) with 400 fake communicating pairs. . . . 93
7.15. Graph obtained considering a malicious node executing the attractive attack in the scattered scenario: (a) with 25 fake communicating pairs and (b) with 400 fake communicating pairs. . . . 93
7.16. Outlier detection using Dixon's test. . . . 94

A.1. Problèmes qui se posent dans les communications sans fil. . . . 104
A.2. Flux de travail. . . . 105
A.3. Modules de détection, de fusion et de présentation : typique architecture du système de surveillance passive. . . . 106
A.4. PePiT user interface. . . . 107
A.5. Distributions des temps d'achèvement des expériences. . . . 108
A.6. DAD: mode d'opération et évaluation. . . . 109
A.7. Tâches supplémentaires proposé pour le module de fusion de traces. . . . 110
A.8. Comparaison des processus de fusion des traces avec le classement proposé et le classement en fonction de la taille des traces. . . . 111
A.9. Comparaison des processus de fusion des traces avec le classement proposé et le classement en fonction du degré des nœuds. . . . 111
A.10. Variation de la métrique strength pour l'attaque répulsive. . . . 113
A.11. Variation de la métrique strength pour l'attaque attrayant. . . . 114

B.1. PePiT communication module UML class diagram. . . . 120
B.2. PePiT communication module UML class diagram. . . . 121
B.3. PePiT communication module UML class diagram. . . . 122

List of Tables

2.1. Summary of variables in opportunistic content dissemination context. 14

3.1. Parameters for the first experiments with PACS and EPICS. 25
3.2. Parameters for nodes of group α, β and γ. 29

4.1. Burst size in function of the node degree. 43

6.1. Trace size for IRCICA and scenarios. 63
6.2. Walktrap modularity values and relative communities for all the similarity metrics. 69
6.3. Infomap modularity values and relative communities for all the similarity metrics. 69
6.4. Label propagation modularity values and relative communities for all the similarity metrics. 69

7.1. Accuracy results obtained with individual and merged traces in both evaluated scenarios. 87
7.2. Cramér-von Mises results for normality hypothesis test. The results above 0.05 do not reject the hypothesis of normality. 94

A.1. Paramètres expérimentaux with PACS and EPICS. 107
A.2. Accuracy obtenue avec les traces individuelles et avec tous les traces fusionnées. 113
A.3. Test de Cramér-von Mises pour l'hypothèse de normalité. Les résultats au dessus de 0.05 ne rejetent pas l'hypothèse de normalité. 114