THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Réseaux, Télécommunications, Systèmes et Architecture

Présentée et soutenue par :

M. GWILHERM BAUDIC

le mardi 6 décembre 2016

Titre :

HINT : DE LA CARACTERISATION DE RESEAU OPPORTUNISTE AU DEVELOPPEMENT D'APPLICATIONS

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Département d'Ingénierie de Systèmes Complexes (DISC-ISAE)

Directeur(s) de Thèse :

M. EMMANUEL LOCHIN

M. TANGUY PERENNOU

Rapporteurs :

M. CONG-DUC PHAM, UNIVERSITE DE PAU ET DES PAYS DE L ADOUR

Mme ANNE FLADENMULLER, UNIVERSITE PIERRE ET MARIE CURIE

Membre(s) du jury :

Mme ANA ROSA CAVALLI, TELECOM SUD PARIS, Président

M. EMMANUEL LOCHIN, ISAE TOULOUSE, Membre

M. PATRICK GELARD, CENTRE NATIONAL D'ETUDES SPATIALES CNES, Membre

M. TANGUY PERENNOU, ISAE TOULOUSE, Membre

Résumé

Les réseaux tolérants aux délais sont aujourd'hui une alternative prometteuse aux réseaux traditionnels basés sur une infrastructure, encore peu déployée. Il existe plusieurs manières d'évaluer les performances d'un tel réseau : expériences de déploiement grandeur nature, modèles théoriques, simulation, émulation, jeu de traces. Chacune a ses avantages et inconvénients, tant en termes de coûts matériels, de réalisme, de temps nécessaire ou de capacité à gérer des noeuds réels. Cependant, aucune ne répond réellement aux besoins des développeurs d'applications.

Dans cette thèse, nous nous focaliserons sur l'émulation. Dans une première partie, nous nous intéresserons aux entrées possibles pour un tel système. Nous proposons tout d'abord un modèle analytique pour prévoir le taux de pertes dans un réseau où les noeuds possèdent une mémoire limitée à un seul paquet. Ensuite, inspirés par les approches de mise à l'échelle de traces de la littérature, nous étudions les hypothèses prises pour l'analyse statistique de traces réelles, et montrons leur in uence sur les lois de probabilité obtenues ainsi que les performances réseau observées. Nous étendons ensuite cette étude à la totalité du cycle de vie des traces réelles, en considérant la collecte de données, le filtrage et la mise à l'échelle de celles-ci. Dans une seconde partie, nous proposons une architecture possible d'un émulateur DTN hybride, c'est-à-dire comportant à la fois des noeuds réels sous forme d'intelliphones, et des noeuds virtuels. Le principal avantage ici est de pouvoir évaluer des applications réelles, éventuellement déjà existantes, dans un contexte DTN, et ce de manière aussi transparente que possible. Nous identifions les limites des approches existantes, ce qui nous permet d'établir une liste de spécifications pour notre système. Nous proposons ensuite un système, nommé HINT, permettant de remplir ces spécifications. L'ensemble est ensuite validé, puis appliqué à l'étude de quelques exemples.

Mots-clés : réseau tolérant aux délais, DTN, émulation, applications opportunistes

Abstract

Delay Tolerant Networks are currently a promising alternative to infrastructure-based networks, but they have not seen a wide deployment so far. There are several ways to evaluate the performance of such networks: field trials, theoretical models, simulation, emulation or replaying contact datasets. Each one has its advantages and drawbacks in terms of material cost, realism, required time or ability to manage real nodes. However, none of them effectively addresses the needs of application developers.

In this thesis, we will focus on emulation. In a first part, we will deal with possible inputs for such a system. We first propose an analytical model to predict the drop ratio in a network where nodes have a one-packet buffer. Then, taking inspiration from trace scaling approaches from the literature, we study the hypotheses and assumptions taken for real traces statistical analyses, showing their impact on the obtained probability distributions and observed network performance metrics. We then extend this study to the whole life cycle of real traces, by considering data collection, filtering and scaling. In a second part, we propose a possible architecture for a hybrid DTN emulator, using both real nodes as smartphones and virtual nodes. The main advantage here is to be able to evaluate real applications, including preexisting ones, in a DTN context, doing so as transparently as possible. We identify the limitations of existing approaches, which helps us build a list of specifications for our system. Then, we propose a system called HINT which matches these specifications. HINT is validated, and applied to the study of some examples.

Keywords: DTN, emulation, opportunistic applications

Acknowledgments

I would like to thank first Prof. Emmanuel Lochin and Dr Tanguy Perennou for allowing me to do my Ph.D. work at ISAE-SUPAERO. Beyond their careful supervision, their comments and support were especially enjoyable.

I am also particularly grateful to Anna Brunström, Johan Garcia and Per Hurtig from Karlstad University for allowing me to do a research stay in their lab, and for the highly useful questionings and remarks we discussed there. This helped us greatly improve the design of our emulation system. Finally, I would like to thank Aaron Clauset, Cosma Shalizi and Laurent Dubroca for releasing the R companion code for the Pareto law on which we based our subsequent developments for statistical analyses. I am also grateful to Sébastien Ardon from NICTA for the helpful discussions and for the original code of the EMO engine, and to Jeremie Bigot for pointing out some issues on the statistical analysis procedures.

I am especially indebted to Prof. Anne Fladenmuller and Prof. Cong-Duc Pham who accepted to review this manuscript, and to Mr. Patrick Gelard and Prof. Ana Cavalli for being the examiners.

I also have to thank all the interns and Ph.D. students in the lab. Unfortunately, I cannot remember all the names and apologize to anyone I may forget here. I am particularly indebted to my coauthors, Victor, Antoine and Khanh, who played a crucial role in developing the analytical model and the emulation system, and helped improve the part on trace analysis. I would also like to thank Rami, Karine, Ahmed, Hamdi, Bastien, Tuan; Nicolas for valuable advice at the beginning of my thesis, Vincent for car repair tricks, Quoc-Viet for highly useful time management techniques, Anaïs for helping me improve my application testing and bug reporting skills but also Alex for his B-movie recommendations (*Return of the Killer Tomatoes* is definitely a must-see), John and Yann for the dietary advice, and Guillaume Smithers Smith for adding a new word to the French language.

This list would not be complete without the GreenMyCity team, a project on urban gardening which we started in November 2015 after a hackathon. Although it did not directly in uence my thesis work, this experience has been truly enjoyable since the very beginning and I sincerely hope it will remain so for the months to come.

Finally, I would like to deeply thank my family for their endless support during all these years.

Toulouse, December 2016

Contents

Acronyms

CCDF Complementary Cumulative Distribution Function

CDF Cumulative Distribution Function

CT Contact Time

CTMC Continuous-Time Markov Chain

DTN Delay-Tolerant Network

DTNRG Delay-Tolerant Network Research Group

DTNWG Delay-Tolerant Network Working Group

ICT Intercontact Time

KS Kolmogorov-Smirnov

ML Maximum Likelihood

TTL Time to Live

Chapter 1

Introduction

Contents

The origins of Delay-Tolerant Networks (DTNs) come from space communications. Due to the high distances considered in this field, delays of several seconds if not minutes are quite usual. Yet, protocols designed for Earth communications are typically designed with timeouts of a few seconds, which makes them inadequate for such uses. But space is not the only field of application for such networks: in fact, they have found an interest in all situations where the network can be disrupted. While the consequence for the end user appears the same (longer delays), the reasons for this are slightly different. Issues range from natural disasters to voluntary interference created by enemies in a war setting, or more simply mobility of the nodes combined with a limited radio range or a limited number of nodes. In the latter case, the favored term is Opportunistic networks, sometimes abbreviated as OppNets in the literature.

Although promising, these networks are currently not deployed. If a killer application for such networks is still to be invented [51], another issue is preventing their deployment: the ability to realistically evaluate their performance, and the performance of applications which would run over them. There are currently four main possibilities for this. Real experimentations are hard to perform, costly and provide low reproducibility. On the contrary, simulations are cheap, very fast and reproducible, but they are made possible by the use of models which simplify the problem. Then, we have analytical models, which aim to predict one specific network performance metric at a very small cost, once the model is derived. However, the amount of simplifying assumptions required to make mathematical derivations feasible represents a major restriction on the use of such models. Finally, emulation brings together real and simulated parts in a same system, thus achieving an interesting trade-off between real experimentations and simulations.

1.1 Contributions of the thesis

The main issue we will study in this thesis is:

How to predict the performance of an application running over a Delay Tolerant Network?

As we mentioned earlier, several solutions to the problem of DTN performance assessment already exist, but they all suffer from various drawbacks and/or limitations. This makes most of them unadapted when it comes to studying the performance of an application running over a DTN network without resorting to a real-life deployment. Indeed, they mostly focus on network performance instead of application performance, which in the case of DTNs is necessary but unfortunately not sufficient.

This thesis consists of two parts. In the first part, we work on possible inputs to our emulation system. We consider a mathematical model for the drop ratio, then focus on real traces, showing the limitations of both approaches. In the second part, the insights gained from the first part are used to design and implement our DTN emulator. We then validate the emulator and apply it to a real case study of application development.

1.1.1 Inputs for a DTN emulator

Before actually designing our emulator, we first need to decide on the type of data we can use for it in order to obtain useful outputs.

To this end, our first contribution is the development of an analytical model for the drop ratio. Analytical models have the benefit of being very easy to use once derived, but often contain a lot of simplifying assumptions to make them mathematically tractable. Here, we study the drop ratio on a network with buffers of one packet, with sources and indistinguishable destinations, while other nodes simply act as relays. We consider two cases for the intercontact distribution: homogeneous exponential and heterogeneous exponential with two classes of nodes.

Next, we deal with real traces, because they appear at first glance to be the most realistic material available for DTN study. Inspired by the EMO work ow [83], our second contribution evaluates the in uence on the statistical fitting results of the assumptions and hypotheses taken beforehand. We find out that some assumptions have bigger consequences than others, and provide a preliminary checklist for practitioners. Then, we try to get a wider view of contact trace usage. We also look at the collection and filtering parts with more attention. Hence, our third contribution aims to list the assumptions taken at the collection, filtering and scaling stages, and illustrate their impact with some examples. Depending on the intended use case for such traces, we then provide a list of recommendations on which assumptions (not) to use, and highlight some of the open questions that remain.

1.1.2 An emulation system: HINT

After this first part related with possible inputs to an emulation system, we move to a second part describing the design and implementation of an emulator for opportunistic

application development. This emulator, called HINT, represents our fourth contribution. Based on the study of the advantages and drawbacks of existing approaches, we provide requirements for such a system, then propose an architecture and a corresponding implementation to fulfill them. Our emulator supports both virtual and real nodes, the latter being Android smartphones. We check that the architecture actually matches the requirements. Finally, we validate and use this emulation system for real case studies using a custom-made opportunistic chat application.

1.2 Organization

This manuscript is organized as follows. It describes different steps towards our emulation platform proposal.

We start by presenting in Chapter 2 the existing approaches for DTN performance evaluation, already summarized at the beginning of this chapter, along with the choice of real traces which will be used throughout this manuscript. This allows us to compare the characteristics of these approaches and justify the advantages of emulation. Chapter 3 is a summary in French of the contents of the manuscript.

The first part of the thesis deals with the preparation of the inputs for an emulation system. Indeed, as is the case with all tools, using correct inputs is a prerequisite to obtaining correct (or, at least, useful) outputs. We first provide in Chapter 4 a mathematical model to study the drop ratio of packets in a particular case of DTN with finite buffers. Next, we dive into the use of contact datasets, also called *traces* in the present work. Chapter 5 discusses their use for statistical analysis and network scaling and the inherent limitations of the process, while Chapter 6 offers a more complete study on the various issues which can arise when experimenting with traces. We do so by considering the complete life cycle of traces, including collection, filtering and scaling. This chapter also provide recommendations for future studies, and lists several open questions that can serve as a starting point for future works.

The second part of the thesis presents the design of our emulation proposal, called HINT. HINT is a lightweight centralized emulation system, specifically aimed towards application developers. As such, it allows to use both real and simulated nodes within a single network, while having at the same time lighter hardware requirements than a full testbed.

In Chapter 7, we will study the limitations of current approaches for performance evaluation on DTNs, and propose a list of requirements to address. This allows us to propose an architecture for an emulation system, along with a corresponding implementation in Python. Then, Chapter 8 uses HINT for various experiments. More precisely, we first validate the ability of our system to effectively address the requirements and provide realistic results. Then, we apply it to a case study of opportunistic application development with a chat application.

Finally, Chapter 9 will conclude the study and provide some perspectives for future work.

Chapter 2

State of the art

Contents

2.1 Delay-Tolerant Networks

The last decades saw tremendous improvements in space exploration. Communication with distant objects, like probes or satellites, is mandatory for their proper functioning, but the very high distances experienced in such conditions require specific protocols to be correctly handled. This was the original context for the appearance of DTNs. However, the ability to deal with delays and/or disconnections is not only valuable for space communications. Indeed, such degraded network conditions can also be experienced on

Table 2.1: State of the art for existing DTN analytical models.

Reference	Routing	Exponential parameter	Memory	Metrics derived			
				Delay	Delivery ratio	Number of copies	Drop ratio
[27]	BSaW	Heterogeneous	Infinite	✓	✓	-	-
[97]	Epidemic	Homogeneous	Finite	✓	-	✓	✓
[74]	Spray and Wait	Homogeneous	Finite	✓	-	-	-
[35]	Epidemic, 2-hop	Homogeneous	Finite	✓	✓	-	-
[1]	SaW, direct	Homogeneous	Infinite	✓	✓	✓	-
[37]	Epidemic	Homogeneous	Infinite	✓	-	✓	-
[44]	Epidemic	Heterogeneous	Infinite	✓	-	-	-
[34]	Epidemic-like, 2-hop	Homogeneous	Infinite	✓	-	✓	-
[38]	Epidemic, 2-hop	Homogeneous	Infinite	✓	-	✓	-
[2]	2-hop	Homogeneous	Infinite	✓	-	✓	-
[72]	Epidemic, 2-hop	Homogeneous	Infinite	✓	-	✓	-
[64]	Simbet, BubbleRap	Homogeneous[1]	Infinite	✓	-	-	-
[79]	Direct	Homogeneous	Infinite	✓	-	-	-
[78]	Spray and Wait	Homogeneous	Infinite	✓	-	-	-
[80]	Custom	Heterogeneous	Infinite	✓	✓	-	-
[10]	Direct, hot potato	Heterogeneous	Infinite	✓	-	-	-
[65]	Epidemic	Heterogeneous	Infinite	✓	-	-	-
[55]	Epidemic, 2-hop	Homogeneous	Infinite	✓	-	✓	-
[82]	2-hop	Not mentioned	Finite	✓	-	-	-
Chapter 4	Hot potato	Homogeneous	Finite	-	-	-	✓

Earth, in the case of natural disasters, military operations, government censorship or poor infrastructure.

To organize the work on this topic, the Delay Tolerant Network Working Group and Research Group were created. The paradigm was also defined more precisely with several RFCs, such as RFC 4838 for the general architecture [86], RFC 5050 for the specification of the Bundle Protocol [77], RFC 6257 for the Bundle Security Protocol and RFCs 7122 and 7242 [45, 26] for the convergence layers to ensure interoperability between the Bundle Protocol and existing TCP and UDP protocols, respectively.

Actual uses of DTNs for terrestrial communications are yet to be determined. However, some suggestions were made in 2009 in [51], while the interest of data o oading for the upcoming 5G cellular networks was investigated more recently in [70].

One necessary condition for the large-scale deployment of such networks is the availability of evaluation tools so their performance can be assessed beforehand. In the remainder of this chapter, we are going to review the main techniques which can currently be leveraged for DTN performance evaluation.

2.2 DTN analytical models

A first possibility suggested by literature is to use models. At the cost of numerous hypotheses and simplifying assumptions, it is indeed possible to derive mathematical models to predict one or more performance metrics of the network being studied. Consequently, several models have been developed in the past years. An overview of their ranges of applicability and provided metrics is given in Table 2.1.

[1] Heterogeneity between nodes is given by contact probabilities instead of intercontacts.

2.2.1 Routing protocols

Due to the wide range of scenarios covered by the DTN paradigm, many routing protocols have been proposed in the literature, with each one targeting a precise use case: some are designed for social networks, while other ones have been proposed with random networks in mind. Protocols can also be classified based on their number of copies, again ranging from 1 for direct transmission to an unrestricted number for epidemic routing. Usually, we differentiate single-copy protocols from multi-copy protocols. A survey of evaluation practices applied to routing strategies was made in [33].

This variety also translates into the protocols used by analytical performance models. We can find 2-hop, Epidemic routing [37, 44, 38], Spray And Wait [1, 78] or Binary Spray and Wait [27], as well as social-based protocols [64], direct transmission [79] or simple forwarding in Chapter 4. Social routing protocols (Simbet, BubbleRap [64]) have been less studied.

2.2.2 Mathematical tools

To our knowledge, the very first paper on this topic [34] introduced a Markov chain model for the number of copies in an homogeneous network of identical nodes, using a two-hop or unrestricted multicopy protocol. It also introduced several assumptions, like exponential intercontact times and instantaneous contacts, which remained in use afterwards. Following this first work, Markov chains remained the most popular tool in the literature for analytical DTN performance modeling.

The main drawback of Markov models is that the state space size can easily explode, either with an increase of the number of nodes or a change in the parameters of the model. An example of such explosion can be found in [27], where increasing the number of copies from 2 to 256 increases the number of states of the chain from 3 to 692005. For extensions to higher numbers of nodes, Ordinary Differential Equations, as an approximated uid limit of Markov models, have been successfully used in [97] or [74]. Recently, Petri Nets were also proposed as a tool to model DTNs [35].

2.2.3 Bu ers

In DTNs, nodes need some storage space to be able to carry the messages from one node to another (recall the Store-Carry and Forward paradigm). The size of these buffers is expected to play a major role in the performance of the network. Yet, many models consider node buffers as infinite to make the mathematical derivation easier. Among analytical models, only a few papers so far have considered finite buffers [97, 74, 35, 82].

With finite buffers comes a need to remove packets when they are no longer useful to the network. For this purpose, the authors of [37] presented 6 schemes of increasing complexity, ranging from a Time to Live (TTL) to a full immunization mechanism, called VACCINE, using antipackets to inform all nodes about the reception of a message and prevent them from accepting the copies remaining in the network. Although Markov models were provided for all the schemes introduced, the Time-to-Live was the one which

gained the most attention in subsequent works [38, 2, 1]. In this scheme, the packet is given a life span expressed in *seconds* (instead of hops in conventional networks) which is passed on at each hop. Once it reaches zero, the packet is erased of all buffers which contain it. While this method is very simple, a poorly adjusted value may prevent the delivery of the message, especially with single-copy routing protocols: the message could expire before even meeting its destination.

Another issue arises with finite buffers: when a new message arrives at a node whose buffer is already full, a choice must be made on which message to drop. The authors of [82] choose to assume that no message will be sent to a node with a full buffer (this policy is called *droptail* in [97]). In [97], two other buffer management strategies are introduced: *drophead*, where the oldest packet is removed and will not be accepted again, and *drophead_sp*, where relayed packets are deleted before source packets when a new packet arrives. Finally, it is also possible to drop packets randomly.

2.2.4 Intercontacts and mobility models

It was shown in [34] that three common DTN synthetic mobility models (namely, Random Waypoint, Random Direction and Random Walk [17]) lead to exponential Intercontact Times (ICTs) when the transmission range is small compared to the network area. This result had a major in uence on subsequent literature; in fact, all papers considered in this survey assume exponentially distributed ICTs, sometimes without explicitly mentioning the underlying assumption on the mobility models. Interestingly, the authors of [46] found that for real traces, the ICT distribution is in fact a power law with an exponential tail. Another paper tried to end this dichotomy between exponential distribution and power law: in [15], the authors showed that the power law characteristic emerged when the nodes never reached the borders of the network area, and was not an intrinsic characteristic of human mobility.

Originally, the distribution was homogeneous, i.e., all pairs had the same parameter for the exponential law. However, this strong assumption is not likely to hold for all networks, and may need to be relaxed. Considering heterogeneity can be done in several manners. First, it is possible to split nodes into groups, as in [55] or [44], and define a parameter for all nodes of each group as well as for inter-group node movements. This will also be the approach chosen in Chapter 4. In such schemes, the distribution parameter is homogeneous inside each group. Full heterogeneity is achieved when each pair of nodes has a different parameter [27, 65]: this can be expressed either directly [27] or through a multiplicative factor applied to a common exponential parameter, such as the contact probability used in [65].

At this point, it is important to note that none of the papers considered here implement heterogeneity with different probability distributions; only changing parameters of a fixed probability distribution is considered. Unfortunately, we will see later in Section 6.5 that different distributions would be necessary to properly capture the characteristics of real contact traces.

2.2.5 Contacts and interferences

In the case when several nodes are in range of each other, interferences are highly likely to occur, thus reducing the available bandwidth. To the best of our knowledge, none of the papers considered in this section considers interference in their model, except [82] where the authors suppose that all other nodes in range remain silent when one is transmitting. The main reason authors invoke to discard interferences is the low density of a DTN, in which the probability of having more than one node within range is likely to be low. This is one of the main hypotheses under the results and formulas presented in [34].

Contact durations are also scarcely taken into account, and are therefore assumed to be instantaneous while allowing to transfer all the data (which equals to having infinite bandwidth, thus making interferences irrelevant). The justification behind this assumption is that contact durations being much smaller than intercontacts, they can be neglected [34]. However, some papers consider finite bandwidths [35]; in [82], contacts allow the transmission of a single message at a time, while in [74] the bandwidth is assumed to follow a known probability distribution.

2.2.6 Node nature

The simplest case would be to consider all nodes as identical. This is indeed the case for most of the models presented here.

To improve the overall performance of the network, several variations of this initial case have been studied. The authors of [37] introduce collection stations, which are fixed nodes used as final destinations for the data collected by sensors. In [44], super nodes are introduced which have a higher speed than the other nodes (e.g., buses among pedestrians), while the authors of [55] assume two classes of nodes with different radio ranges. Interestingly, the heterogeneity of nodes will have to be re ected in the ICT [10, 65] and/or Contact Time (CT) distributions.

2.2.7 Tra c model

The simplest traffic model is to have a single source-destination pair exchanging a unique packet [27, 34, 44, 38, 2, 72]; the rest of the network is only used for relaying. This approach has the advantage of being easier to model mathematically. Authors also claim that once the models are derived for this simple case, extension to more complex schemes is straightforward. A step further is to consider data ows (several packets between a source and a destination), either for one [74] or several [97] source-destination pairs. In [74], the authors also envisioned the case of additional unrelated traffic in the network.

2.2.8 Derived results

Since all models studied here aim to give insight on the performance of networks, it is therefore important to present the typical performance metrics considered in the study of

DTNs. With an acronym explicitly mentioning delays, it is not surprising that almost all papers, regardless of the buffer size considered, aim at deriving the delay of the network, which is the time for a packet to go from the source to the destination.

For multi-copy routing protocols, the number of copies has also been studied; apart from the total buffer occupancy in the network, another interest of this measurement comes from its close relationship to the energy consumption of the nodes. For resource-constrained networks such as DTNs, energy is a quantity whose role cannot be underestimated.

Other metrics include the number of hops needed to go from the source to the destination, the delivery ratio (number of packets which reach the destination within a given time frame) or the drop ratio (number of packets lost due to buffer over ow or expiration).

For simplicity, we did not distinguish between papers providing the average value for the metrics and those giving the complete probability distribution. Having only the average value for a given metric is unlikely to be sufficient for application development purposes, because it does not give any information on the range of possible values. Another major drawback of such models comes from the hypotheses used. Indeed, relaxing only one of them (changing routing protocol, for example) is sufficient to render the whole model useless. It is also noteworthy that all these models rely on the assumption of exponential intercontacts introduced by R. Groenevelt in [34]. While very practical from a mathematical standpoint, this finding was done on the output of synthetic mobility models (random waypoint, random direction and random walk [17]) and may not hold for other models or for actual human mobility.

2.3 Real traces

Speaking of human mobility, a second way is to use traces captured in diverse settings (rollerskating tour, conference, mobile apps usage...), such as those archived on the CRAWDAD[2] portal. Beacuse the goal of this thesis is not to perform a comparison of all traces available for DTN study (nearly 20 in CRAWDAD at the time of writing), we chose to focus solely on four datasets. These will be subsequently used in the remainder of this work, especially Chapters 5 and 6. An overview of their characteristics is provided in Table 2.2.

The first dataset is **Rollernet** [5], which was collected during a 3-hour long rollerskating tour in Paris in 2006. 62 Bluetooth contact loggers (iMotes) were distributed to volunteers and staff members among approximately 2,500 participants. Among the participants of the experiment, some had known locations (front, rear, side...), while others were free to move within the crowd.

The second one comes from the **Infocom 2005** experiment [76], part of the larger Haggle project (5 experiments), which also relied on similar contact loggers. They were distributed among 41 participants of the student workshop of the conference, who also attended the rest of the conference afterwards for a total duration of three days.

[2] http://crawdad.cs.dartmouth.edu

Table 2.2: Main characteristics of the datasets used in this work.

	Rollernet	MIT	Infocom '05	Humanet
Technology	Bluetooth	Bluetooth	Bluetooth	Bluetooth
Device	iMote	phone	iMote	custom
Environment	urban	campus	conference	office
Duration (days)	0.125	284	3	1
Time span	NA	24/7	24/7	workday
Sampling period (s)	15	300	120	5
Internal nodes	62	89	41	56
Internal contacts	60,146	114,046	22,459	64,445
External contacts	72,365	171,466	5,757	64,531

The third dataset, **MIT Reality Mining** [28], was collected through the use of an activity logging application embedded in mobile phones. These were lent to 100 MIT students from several departments (in computer science and business) and several study years over the course of the 2004-2005 academic year, representing almost 9 months of data. For this third dataset, we only considered the Bluetooth contact traces in our studies (thus ignoring all information on the cellular network), and restricted the data to the 89 devices which effectively recorded contacts. To avoid exhausting the memory too fast, the sampling period was chosen by the authors to be quite large (300 s).

Finally, the **Humanet** dataset [12] was also produced using custom hardware (called PDPD) and Bluetooth. It recorded contacts between 56 people in an office environment of a research lab in Spain during workdays (no nights or weekends) for 6 weeks in 2010. However at the time of this writing, only 1 day has been published so far. We only considered contacts recorded between users when the devices were worn, thanks to the use of the mobility flag recorded in the trace. The major interests of this trace are twofold. First, the data is more recent than the other datasets chosen here. Secondly, and maybe more importantly, the choice of parameters like the time span and sampling period was made after a careful study [13]. We will come back to it later in Chapter 6.

As can be seen from the previous data, the scope covered by these four traces is quite broad, be it in terms of node number, time span, year of collection, environment or sampling period. On the contrary, it can be noted that all are using Bluetooth for the radio technology. The number of nodes is in all cases quite small (below 100), which is inconvenient to study large scale systems. For completeness, we have to mention that there are also other collection efforts using network packet logs, or WLAN access point records, which have been used in the DTN literature. However, because those traces do not especially target DTNs, a common assumption for WLAN traces is to consider two nodes as in contact when they are both simultaneously connected to the same access point. Note that it may not always be the case in practice, especially if shorter-range technologies are used.

2.4 Simulation

Network simulators are a highly popular approach to get a first impression on performance of new algorithms or applications. For DTN and opportunistic networking, the main tool is the ONE [47], written in Java. It is however also possible to use more generic simulators like ns-3 [39] or OPNET [21]. The main advantages of simulators are their low cost and ease of use, coupled with a high speed: for example, an experiment on a 5000 second dataset can take as little as 7 seconds to complete with the ONE. The main drawback is the lack of support for real applications.

Being especially tailored to DTN study, the ONE provides several routing algorithms and movement models. External traces as presented in Section 2.3 are supported as well, at the cost of a simple format conversion. It also offers report classes for the major DTN metrics like delivery ratio, number of hops or message delay. If necessary, other reports can be produced by writing new classes.

More recently, a new simulator called Adyton [60] has been proposed by a team of the University of Ioannina in Greece. Written in C++ for Linux, this simulator does not include any mobility models but relies instead on contact traces as inputs. A very large choice of routing protocols, buffer management policies or congestion control mechanisms is offered. However, the only input method available is to use traces, which have to be preprocessed by a customized script to become usable. This script performs some of the filtering steps we will describe later in Section 6.4. Also note that a different script has to be produced for each trace: if a new trace is published, it will require as well a corresponding script to become usable in Adyton.

2.5 Emulation

It can sometimes be hard to differentiate between emulation and simulation. A comparison of both was made for example in [53]. According to [7], *network emulation is a network experiment technique that employs an experimental setup containing both real network components, be it hardware or software, and components that are reproduced virtually through computer models.* As such, it can be seen as an intermediate between pure simulation and real deployment, taking some elements from each side and putting them together. For example, network nodes can be setup as *virtual* machines executing a *real* DTN protocol stack like DTN2 [25], ION [11] or IBR-DTN [75]. It is also possible to run real machines on a virtual network.

In this section, we will discuss two sides of emulation. The first one aims to reproduce the whole network by using a machine for each node. The second one will discuss link-level emulators, which are pieces of software used to apply impairments to a single link between two machines, for example inside a testbed.

2.5.1 Testbeds

The simplest approach that we can envision is to use a real machine for each node in the network. Most of the time, such machines are virtual for scalability reasons. All these

machines are then put together in a same facility, and form what is called a *testbed*. However, such systems require a lot of hardware and are especially difficult to setup; consequently, an important part is played by the management software, such as OMF.

Existing systems are numerous. Examples include QOMB [6, 8], TUNIE [50] or MoViT [32]. In the On/Off-based mobility emulator from [95], mobility is replayed by moving the applications from one node to another. HYDRA [57] uses virtualized instances of OpenWRT, and replays contact traces between the hosts to reproduce connections. A similar approach is also taken in [9].

Hybrid emulation is commonly defined in the literature as the association of both real and emulated/simulated nodes in the same studied network. The main advantage of this approach is its scalability, thanks to the use of simulation for a part of the network. Indeed, above solutions only use real software, other elements being virtualized. For instance, in TWINE [99], simulated, emulated and real nodes interact in the same testbed. The TROWA testbed [54] relies on a discrete event simulation and also binds real and simulated nodes. The authors of [52] chose to use real machines communicating over a synthetic network, with a VPN solution being used to apply degradations to the experimental network.

In [8], the authors provide some design guidelines to include real nodes in an existing testbed and propose a solution to connect these additional nodes to both the experimental and control networks from the testbed. Another proposal from the literature is [98]. In this work, the authors use a central computer to emulate a DTN with several nodes, while both ends of the network are running on two other computers. A similar architecture will be presented in Chapter 7.

Most of the time, physical hosts are located in the premises of the research lab. Some proposals like [32] have suggested to use commercial cloud services such as those provided by Amazon. This makes large scales much easier to attain, at the expense of some additional constraints in terms of synchronization and delays.

In the present work, we want to abstract the network more and, ideally, envision it as a huge node connected between a source and a destination.

2.5.2 Link-level emulators

If we take a closer look, it may be necessary to produce some impairments at the link scale. Such a feature can be useful both in a testbed, to disrupt network links between nodes, or in the simpler case of a unique node mentioned above. For this purpose, several link emulators exist in the literature. An overview of their features is given in Table 2.3. Note that they are not especially aimed at DTN systems.

Netem Netem is a standard Linux component. By specifying rules on network interfaces, one is able to create various impairments (delay, losses, bandwidth, duplication, reordering...). This is achieved through the use of queueing disciplines.

Table 2.3: Feature comparison of various link-level emulators.

Name	Netem	Dummynet	KauNet	IMUNES
Latest version	-	2013	2010	2016
Delay	✓	✓	✓	✓
Losses	✓	✓	✓	
Bit errors	✓	-	✓	✓
Bandwidth	✓	✓	✓	✓
Reordering	✓	-	✓	-
Triggers	-	-	✓	-
Duplication	✓	-	-	-

Dummynet Dummynet [18] is a standard FreeBSD component, also available for Windows and Linux. It uses the concept of pipes to intercept network traffic and apply various impairments as needed, including bandwidth changes, delay or packet losses.

KauNet and KauNetEm KauNet is originally a FreeBSD module developed at the University of Karlstad [62], with also a Linux port. It is an extension of Dummynet. One of its main distinctive features is the use of *triggers*, which can be useful to create cross-layer effects like connections or disconnections in the context of DTNs [62]. Recently, an updated version called KauNetEm [31] has been developed. It brings most of the features of KauNet to Linux because it tends to be more popular as FreeBSD, and is based on Netem instead of Dummynet. The main addition compared to Netem is a deterministic placement of the perturbations. Consider for example a loss rate of 2%: in this case, Netem will drop on average 2 packets every 100 packets, without any control on the exact value and of the exact positions of the packets lost. This lack of sufficiently precise control can be an issue for some experimenters, and is one of the motivations behind the development of KauNetEm.

IMUNES Developed at the University of Zagreb, IMUNES [96] is a network emulator/simulator running on FreeBSD. It also offers a GUI to setup experiments.

We notice that all these systems require as inputs the conditions we wish to emulate. In the DTN case, delay and delivery ratio (which can indirectly translate to a drop ratio) seem absolutely necessary. Bandwidth changes can also prove useful, for example to model interferences caused by other nodes transmitting at the same time.

The question which gets asked is how to go from the initial network to the metrics required for emulator operations. A first possibility is to play real contact traces or synthetic scenarios in a simulator like the ONE [47] or SWINE [24], and to collect the output results. However, this approach depends from the limits of the simulator, especially in terms of node number, supported features or total memory footprint. Another possibility is to use the simplified analytical models presented in Section 2.2 to derive the required metrics. This approach is more exible than simulation, but we saw that these models required several assumptions to become mathematically tractable. Depending

on the metrics needed and on the various characteristics of the studied network (routing, mobility type, buffer size...), using a model may be impossible, simply because one does not exist for this precise network setup.

2.6 Discussion

A comparison of the different approaches for network performance evaluation, although not especially targeted to the study of DTNs, can be found in [7]. For the reader's convenience, we reproduce it here in Table 2.4.

Table 2.4: Comparison of network performance assessment methods (from [7]).

	Analytical models	Simulators	Real experiment	Emulation
Cost	Very low	Low	High	Medium
Real time	NA	No	Yes	Yes
Control on conditions	Very high	High	Low	High
Reliability	Very low	Low	High	Medium
Ease of use	Very high	High	Low	Medium

From the table, emulation appears as a good trade-off for all criteria considered. Furthermore, it allows to operate in real time and accepts real applications.

The problem of current approaches is that they mostly aim to provide network characterization. Although valuable, this is not the primary focus for an application developer: the main interest in this case is to be able to assess the proper functioning of the software. This requires a tool on which real devices (and consequently real applications) can be easily plugged in. Because they are not designed for it, both analytical models and simulators can be ruled out. Real experiment is also a poor candidate, due to its high cost, low repeatability and inconvenient deployment in an existing development work ow.

For this reason, we choose to focus on emulation in the remainder of the thesis. However, as was pointed out in this chapter, every emulation system needs inputs to be able to recreate network conditions. Hence, we will now study in the next chapters two possible ways of obtaining inputs for our emulation system.

Chapter 3

Résumé de la thèse

Contents

Dans ce chapitre, nous présentons un résumé de la thèse en français. Par simplicité, l'ordre des chapitres et des parties a été conservé. Les liens vers les références bibliographiques se trouvent dans la version anglaise. Lorsque c'est nécessaire, nous renvoyons pour plus de détails aux sections correspondantes dans la suite du manuscrit.

3.1 Introduction

Les réseaux tolérants aux délais sont une alternative prometteuse aux réseaux actuels basés sur une infrastructure, malheureusement encore très peu déployée. S'il existe plusieurs manières d'évaluer les performances de tels réseaux (test réel, modèles analytiques, simulation, émulation. . .), aucune ne répond réellement à ce jour aux besoins des développeurs d'applications. Le besoin d'intégrer des noeuds réels dans le réseau nous conduit dans le cadre de cette thèse à nous intéresser plus particulièrement à l'émulation.

Le manuscrit (et par conséquent ce chapitre) se décompose comme suit. Dans un premier temps, nous nous intéressons aux entrées possibles pour un système d'émulation. Nous commençons dans la Section 3.2 par introduire un modèle analytique du taux de pertes dans un cas particulier de réseau. Nous évoquons ensuite plus en détail le cas des traces réelles, que ce soit pour l'analyse statistique (Section 3.3) ou pour leur cycle de vie complet, de la collecte à la mise à l'échelle (Section 3.4). Ceci nous permet de formuler des recommandations pour de futures études. Une fois ce travail sur les entrées effectué, nous

passons à la définition de l'émulateur proprement dit. Nous définissons les spécifications et l'architecture correspondante dans la Section 3.5, et proposons une implémentation en langage Python. Enfin, dans la Section 3.6, nous validons notre émulateur dénommé HINT et l'appliquons à l'étude d'une application de *chat* opportuniste.

3.2 Un modèle analytique : taux de pertes dans les réseaux DTN avec une mémoire à un seul paquet

La plupart des efforts de modélisation analytique des DTNs s'attachent au délai ou au taux de délivrance, l'occupation mémoire étant généralement considérée comme secondaire avec l'emploi usuel de mémoires de taille infinie. En conséquence, le taux de pertes apparaît comme l'un des parents pauvres des modèles analytiques. Dans cette section, nous nous attachons au taux de pertes dans un réseau à mémoires finies, via une modélisation par chaîne de Markov. Nous étudions le pire cas avec des mémoires limitées à 1 paquet, dans le cas d'un réseau à intercontacts homogènes puis hétérogènes à 2 classes. Notre contribution ici est de lier lois d'intercontact et taux de pertes. Le modèle mathématique est vérifié par des simulations.

En raison de la nature particulière des réseaux DTN, de nombreux travaux se sont intéressés à leurs restrictions caractéristiques, puis à la prédiction des performances, comme présenté en Section 2.2. Dans ce domaine, les chaînes de Markov ont été utilisées très tôt, et ce pour de nombreuses configurations de routage. D'autres outils exploités plus récemment incluent les réseaux de Petri ou les équations différentielles. Cependant, la quasi-totalité des modèles existants font abstraction de la mémoire en la considérant comme infinie. Dans cette section, nous nous intéressons à un type particulier de réseau de capteurs dans lequel des nœuds mobiles effectuent des mesures et les transmettent via les autres capteurs. On considère 2 groupes : les sources qui effectuent les mesures, et des stations de collecte qui sont les destinations. Les autres noeuds ne servent que de relais. Toutes les destinations sont considérées comme équivalentes du point de vue du routage. Nous étudions l'évolution du routage et les pertes de paquets lorsque le nombre de messages générés augmente. Le routage utilisé ici est sans réplication (*simple forwarding*) pour éviter d'introduire de nouvelles copies, donc potentiellement des pertes supplémentaires. L'étude est limitée au cas où les nœuds ne peuvent contenir qu'un seul message, qui peut aussi être vu comme la limite avant saturation de la mémoire et par conséquent perte de message. A cet effet, nous proposons une chaîne de Markov à temps continu pour caractériser les pertes de messages, dans l'objectif de dériver une borne supérieure au taux de pertes. De là, il est possible de trouver un compromis entre nombre de sources et de destinations pour conserver un taux de pertes borné. Les résultats du modèle théorique sont validés par la conformité avec les simulations. Nous dérivons le modèle pour deux cas : un réseau entièrement homogène, puis un réseau hétérogène avec 2 classes de nœuds.

On considère un réseau DTN avec N nœuds identiques, chacun avec une capacité de 1 message (Section 4.2). Parmi ceux-ci, on considère S sources et M messages (avec $S = M$ compte tenu de la restriction sur la mémoire). Parmi les $N - S$ nœuds restants,

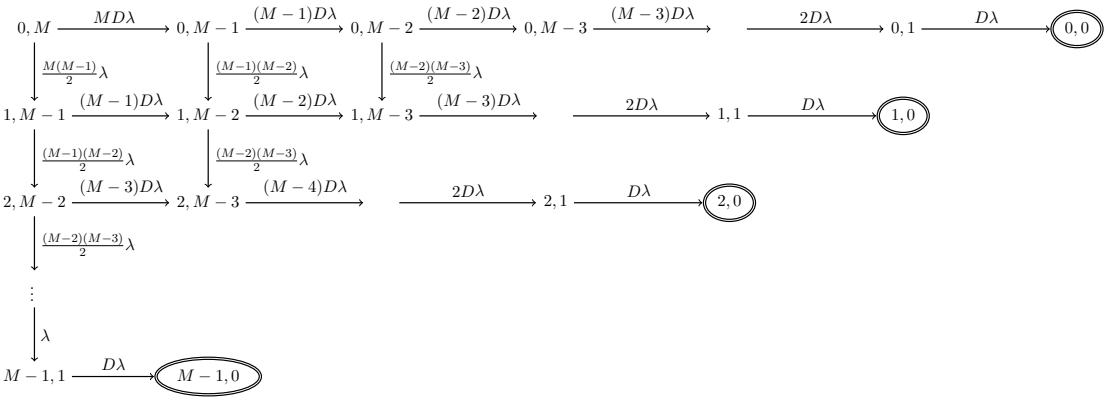


Figure 3.1: Représentation de la chaîne de Markov dans le cas homogène.

D sont des destinations, le reste sert de relais. Sauf mention contraire, on considérera que $D = 1$. Les nœuds intermédiaires se contentent de faire suivre les messages, sans effectuer de copies. On suppose enfin que l'échange de messages est instantané. A chaque contact, chaque nœud essaie de transmettre son message. Il y a donc 2 cas : soit un seul des 2 nœuds a un message, et celui-ci est transmis ; soit les 2 nœuds ont déjà un message, alors un nœud choisi aléatoirement transmet son message et l'autre message est perdu.

Dans le cas homogène, on suppose que les intercontacts de chaque paire sont indépendants et identiquement distribués selon une loi exponentielle de paramètre $\frac{1}{\lambda}$. Pour calculer le nombre de messages perdus, on introduit une chaîne de Markov dont les états sont notés (d, c), où d est le nombre de messages perdus et c le nombre de copies dans le réseau. L'état initial est noté $(0, M)$. On peut sortir d'un état (d, c) donné de deux manières : soit vers $(d, c-1)$ lorsqu'un message est reçu ou vers $(d+1, c-1)$ lorsqu'il y a perte. Le taux de rencontre avec une destination est $D\lambda$. Comme il y a c nœuds avec un message, le taux pour cette transition est donc $cD\lambda$. Les états absorbants de la chaîne sont de la forme $(d, 0)$ avec $0 \leq d \leq M-1$. Il existe cependant certaines restrictions sur les états absorbants : en effet, comme le dernier message sera reçu avec une probabilité de 1, il est impossible de passer de l'état $(d, 1)$ à $(d+1, 0)$. La chaîne obtenue est représentée en Figure 3.1. On peut remarquer dans les formules des probabilités de passage d'un état à un autre que celles-ci sont indépendantes du paramètre de la loi d'intercontacts.

Nous étendons à présent (en Section 4.2.3) le modèle homogène (d, c) à un cas avec 2 classes de nœuds $\mathcal{C}_1$ et $\mathcal{C}_2$ telles que $\mathcal{C}_1 = N_1$ et $\mathcal{C}_2 = N_2$ avec $N = N_1 + N_2 + D$. Par simplicité, nous imposons $D = 1$. De plus, la destination se trouve en dehors des 2 groupes de nœuds, pour garder le problème symétrique. On a $M = M_1 + M_2$ messages dans le réseau, M_1 dans $\mathcal{C}_1$ et M_2 dans $\mathcal{C}_2$. L'hétérogénéité est donnée par l'usage de paramètres de loi exponentielle différents dans chacun des groupes : $\frac{1}{\lambda_1}$ dans $\mathcal{C}_1$, $\frac{1}{\lambda_2}$ dans $\mathcal{C}_2$ et $\frac{1}{\lambda}$ pour les interactions inter-classes. Ces différences de valeurs permettent d'affecter des vitesses de rencontre des nœuds différentes selon les groupes, par exemple

avec un groupe avec délivrance plus rapide (donc moins de pertes). Par symétrie, la destination rencontre les nœuds de $\mathcal{C}_1$ avec un paramètre $\frac{1}{\lambda_1}$ et ceux de $\mathcal{C}_2$ avec un paramètre $\frac{1}{\lambda_2}$. Cette restriction peut être levée et n'affecte pas le résultat général. Ici, les états sont notés (d, c_1, c_2) avec d le nombre de pertes comme précédemment, c_1 le nombre de copies dans la classe $\mathcal{C}_1$ et c_2 dans la classe $\mathcal{C}_2$. 3 types de transitions sont possibles ici : réception de paquet à la destination, pertes de paquets ou passage de paquets d'une classe à une autre. Comme précédemment, il y a M états absorbants de la forme $(d, 0, 0)$ avec $0 \leq d \leq M-1$. Il est possible d'arriver à l'état absorbant depuis $(d, 1, 0)$ avec un taux λ_1 ou depuis $(d, 0, 1)$ avec un taux λ_2. Là encore, certaines transitions ne sont pas possibles : par exemple, il est impossible de passer de (d, N_1, c_2) à $(d, N_1 + 1, c_2 - 1)$ ou de $(d, 0, c_2)$ à $(d, -1, c_2 + 1)$.

Nous présentons à présent (Section 4.3) les résultats de notre comparaison entre le modèle markovien et un réseau simulé. Les intercontacts sont générés comme expliqué précédemment, puis ces événements sont joués dans un simulateur orienté événements. Le modèle est calculé avec MATLAB, et la simulation est implémentée en R. Pour toutes les simulations, $N = 100$, et la mémoire est d'un seul message. Chaque simulation est répétée 10 fois pour donner un intervalle de confiance à 95 %. On définit aussi l'occupation du réseau π comme le rapport nombre de sources/nombre de nœuds total. Le nombre de sources est ensuite augmenté pour balayer les différentes valeurs possibles. Dans le cas entièrement homogène à une seule destination (Section 4.3.1), on voit clairement l'augmentation rapide du taux de pertes avec l'augmentation du nombre de sources. La différence constatée entre le modèle et les simulations est de $0,04$ en moyenne. Avec la variance, la différence sera plus importante, notamment avec un faible nombre de sources. Avec plusieurs destinations (nombre variant de 1 à $N - S$), on constate clairement en Section 4.3.2 la réduction du taux de pertes avec l'augmentation du nombre de destinations. Là encore, modèle et simulation sont en accord. L'apport de cette étude est de pouvoir définir le nombre de nœuds nécessaires pour limiter le taux de pertes : par exemple, avec 20 destinations le taux de pertes reste inférieur à 25 % pour une occupation réseau inférieure à 35 %. On peut aussi définir le nombre maximal de sources admissibles pour garder un taux de pertes borné.

Enfin, dans le cas hétérogène à 2 classes (Section 4.3.3), nous retenons $N_1 = 50$, $N_2 = 49$, $D = 1$. Deux expériences sont effectuées, avec des valeurs de paramètres exponentiels différents. Dans les 2 cas, le modèle homogène a de moins bonnes performances que les modèles hétérogènes. Notez que le cas hétérogène n'est pas indépendant des paramètres de lois, contrairement à l'homogène. Les deux expériences hétérogènes donnent d'ailleurs des taux de pertes différents, ce qui s'explique par des choix de paramètres différents en termes d'ordres de grandeur. Encore une fois, modèle et simulations sont en accord.

Nous présentons ensuite brièvement quelques extensions possibles du modèle, en considérant des mémoires plus grandes et/ou davantage d'hétérogénéité (Section 4.4). Si celles-ci sont théoriquement faisables, il convient de noter que l'évolution du nombre d'états nécessaires conduit à une explosion dans chaque cas.

Dans cette section, nous avons étudié le taux de pertes dans un réseau composé de nœuds sources et de nœuds destinations, où chaque source émet un message pouvant

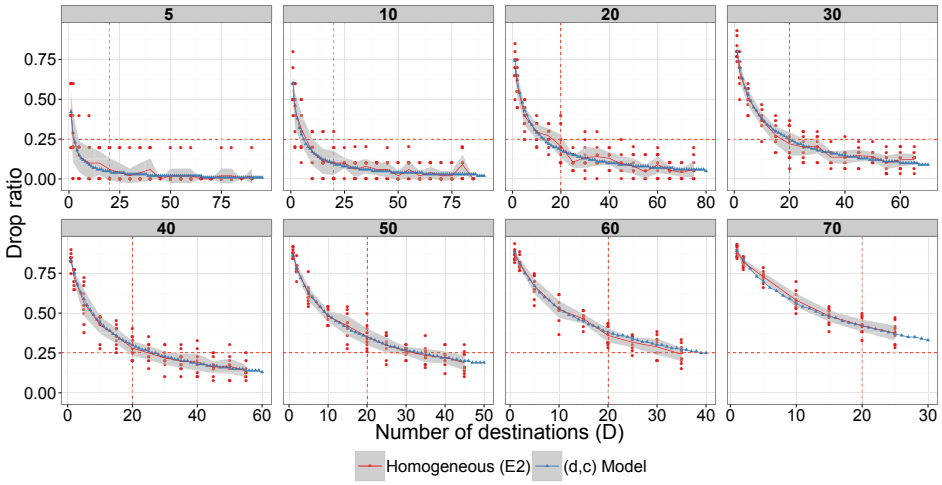


Figure 3.2: Evolution du taux de pertes pour différents taux d'occupation du réseau et nombres de destinations.

être reçu par n'importe quelle des destinations. Nous utilisons un routage sans réplication pour éviter la création de nouvelles copies. Le cas des mémoires à un seul paquet est celui d'une quasi-saturation du réseau. Nous proposons un modèle markovien pour caractériser le taux de pertes dans ces conditions, dans 2 cas : un réseau complètement homogène (modèle (d, c)) et un réseau hétérogène à 2 classes (modèle $(d, c1, c2)$). Ce modèle nous permet de montrer le lien entre lois d'intercontacts et taux de pertes observés. Il nous permet aussi de trouver un compromis entre nombre de sources et de destinations pour conserver un taux de pertes borné. Le modèle est validé par comparaison avec des simulations. Pour de prochains travaux, nous envisageons de modifier les paramètres des lois d'intercontact du modèle à 2 classes, mais aussi de considérer les cas avec des mémoires plus grandes ou une totale hétérogénéité.

Il convient à ce stade de rappeler que les modèles analytiques comme celui que nous venons d'introduire souffrent de plusieurs limitations. Tout d'abord, leur dérivation n'est rendue possible que par le choix de nombreuses hypothèses et simplifications préliminaires (contacts instantanés, algorithme de routage...). Il suffit parfois d'en relaxer une seule pour que le modèle proposé devienne inutilisable. Ensuite, le nombre de métriques réseau fournies par un tel modèle est généralement limité à une ou deux, comme le taux de pertes dans notre cas. L'accès à une seule métrique, bien qu'utile, n'est cependant pas suffisant pour une étude correcte. Pour tenter de passer outre cette limitation, nous nous penchons maintenant sur les traces réelles.

3.3 Analyse statistique de traces réelles

Les traces de contacts collectées lors d'expérimentations réelles représentent un matériau populaire pour l'évaluation des performances d'un DTN. Ces traces nécessitent générale-

ment un prétraitement pour être pleinement utilisables. Cependant, plusieurs hypothèses sont souvent prises avant de procéder à l'analyse statistique des contacts et les temps d'inter-contacts. Dans cette section, nous classons tout d'abord ces hypothèses, et analysons leur impact sur la caractérisation statistique de trois jeux de données couramment utilisés au sein de la communauté de recherche en DTN. Nous identifions également certains pièges dans l'analyse de données qui pourraient fortement in uencer les conclusions faites par l'expérimentateur. Sur la base de notre propre expérience, nous proposons par la suite une liste préliminaire pour aider les chercheurs à éviter ambiguïtés ou malentendus indésirables dans leurs études.

En effet, les traces de contact ont été très souvent utilisées par des chercheurs souhaitant apporter une touche additionnelle de réalisme à leurs études. La production de traces étant un processus coûteux et contraignant, seul un petit nombre existe, ce qui fait que ce sont souvent les mêmes jeux de données qui sont utilisés pour différents articles. Face à cela, des approches de mise à l'échelle par traitement statistique ont été proposées, pour permettre de modifier le nombre de noeuds et/ou la durée de traces existantes. Cependant, le réalisme obtenu avec ces ensembles de données est discutable.

Pour cette étude, nous utilisons 3 jeux de données : Infocom 2005, MIT et Rollernet. Ceux-ci ont l'avantage d'être parmi les plus anciens jeux de données à notre disposition, et donc parmi les plus utilisés dans la littérature. Après avoir détaillé les différents outils statistiques possibles, nous choisissons le test Kolmogorov-Smirnov (KS). Nous listons ensuite les choix et hypothèses prises dans la littérature existante avant une analyse statistique de traces de contact (Section 5.2.3). Ils sont au nombre de 7. Tout d'abord, le choix des noeuds a son importance : avec ou sans noeuds dits "externes", qui ne participent pas à la collecte mais apparaissent néanmoins dans la trace. On peut aussi jouer sur la symétrie des contacts, le nombre minimum de contacts par paire, la durée totale de la trace ou la définition de l'inter-contact. Il est également possible de considérer ou non les très nombreux contacts de durée nulle. Enfin, il est possible de choisir pour estimer les paramètres de la loi de Pareto entre 2 approches distinctes, à savoir un choix arbitraire ou un algorithme d'estimation à partir des données.

Nous présentons ensuite les conséquences de ces hypothèses sur les résultats de l'analyse statistique (Section 5.3), en nous concentrant sur les temps de contact agrégés. Nous avons testé trois distributions statistiques : exponentielle, log-normale et Pareto, qui sont les plus représentées dans la littérature. Les outils utilisés sont le test KS, des estimateurs du maximum de vraisemblance pour les paramètres des distributions statistiques et le logiciel R. Parmi les 7 hypothèses évoquées, nous en étudierons 5, les deux autres ne s'appliquant pas au cas considéré ici.

Nous nous intéressons tout d'abord aux contacts de durée nulle (Section 5.3.1), avec trois possibilités : suppression, extension à 1 seconde (durée arbitraire), ou suppression de toutes les valeurs inférieures à la période d'échantillonnage. Les variations observées dans deux exemples sont très importantes : dans un cas, reproduit en Figure 3.3, on obtient même un changement de distribution statistique. Ceci peut notamment s'expliquer par la proportion importante de contacts de durée nulle dans toutes les traces utilisées. Ces contacts sont d'ailleurs paradoxaux : ils apparaissent dans les traces, ce qui im-

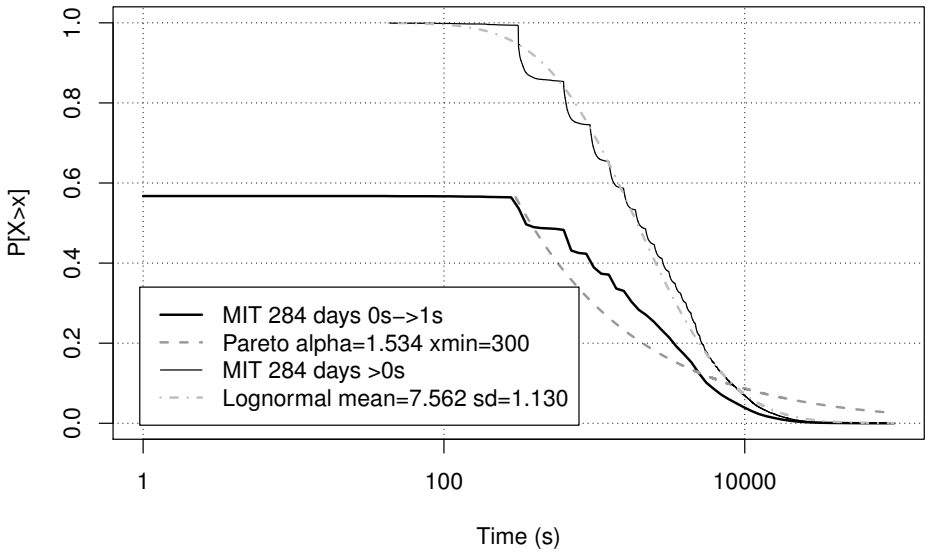


Figure 3.3: In uence des contacts de 0 seconde dans le cas de la trace MIT.

plique une transmission de données, alors que la durée nulle empêche toute possibilité de transmission. Ce paradoxe sera étudié plus en détail au Chapitre 6. Nous pensons donc qu'étendre les contacts à une durée arbitraire comme 1 seconde est le meilleur compromis entre transmission de données et court temps de contact.

Concernant la loi de Pareto (Section 5.3.2), deux techniques d'estimation de paramètres sont possibles : utiliser la période d'échantillonnage, ou avoir recours à l'algorithme de Clauset et al. En comparant ces deux approches dans le cas du jeu de données Infocom 2005, il est clair que ce changement a un impact majeur. Bien que les résultats fournis par ces deux approches ne soient en réalité pas contradictoires, ne capturer qu'une partie des données (qui plus est, minime dans ce cas) est problématique dans la mesure où le traitement à réserver aux données restantes n'est pas déterminé. Il peut être intéressant dans certains cas de ne prendre qu'une partie de la trace, pour enlever des périodes pouvant présenter des propriétés ou comportements différents, comme des fins de semaine ou des vacances scolaires. Pour deux traces différentes (MIT et Rollernet), nous obtenons en Section 5.3.3 un faible impact sur les résultats, et ce sur les CTs comme les ICTs. La présence ou non des noeuds externes (Section 5.3.4) n'a qu'une in uence limitée sur les résultats. Notons que des conclusions similaires avaient déjà été trouvées auparavant, mais pour un autre jeu de données collecté dans un environnement différent. Ceci semble indiquer que noeuds internes et externes présentent des comportements proches, quelles que soient les conditions expérimentales. Enfin, la définition alternative de l'intercontact (Section 5.3.5) a également une importance limitée en termes d'impact sur les intercontacts ou sur les distributions associées. En revanche, l'impact d'une telle hypothèse sur

les contacts et/ou la longueur totale de la trace ne doit surtout pas être négligé.

Nous proposons à présent une première liste de critères à vérifier lors d'une analyse statistique, sous forme de 3 questions (Section 5.4). Le but est de rendre les auteurs conscients que des modifications apparemment anodines peuvent en fait avoir des conséquences plus importantes que prévu sur les résultats ultérieurs. Premièrement, on indiquera si des filtres sur les valeurs ou les périodes de temps (vacances, par exemple) ont été appliqués, et dans quelle optique. On vérifiera également si des choix ultérieurs, comme les algorithmes d'estimation, n'effectuent pas aussi un filtrage supplémentaire sur les données. Enfin, il convient également d'indiquer si au delà du filtrage, certaines valeurs ont dû être changées (cas des contacts de 0 seconde avec des estimateurs Maximum Likelihood (ML), par exemple).

A ce stade, il convient de mentionner que l'analyse statistique n'est en fait qu'une étape vers un objectif final qui est de pouvoir prédire les performances du réseau. Aussi, nous nous intéressons à présent (Section 5.5) à l'in uence sur les performances réseau (délai et taux de délivrance) des hypothèses étudiées précédemment. L'étude est limitée à 3 hypothèses : contacts de 0 seconde, longueur de la trace et noeuds externes, les autres étant indépendantes de l'analyse de performance. Nous nous limitons également aux deux traces les plus courtes, Infocom 2005 et Rollernet. Pour les contacts de 0 seconde, l'in uence est différente selon la trace considérée : inexistante pour Infocom, importante pour Rollernet, ce qui était attendu. La longueur de trace, étudiée uniquement pour Rollernet au vu des choix pris dans la littérature, a un impact faible. Enfin, l'inclusion des noeuds externes ne présente aucun intérêt dans le cas Infocom mais permet d'améliorer les performances sur Rollernet, en raison du choix de contacts symétriques fait par les auteurs. On constate également que les conclusions de la partie précédente ne s'appliquent pas forcément aux performances. Nous évoquons ensuite brièvement quelques pistes pour des expériences supplémentaires, notamment en modifiant certains paramètres.

Nous venons d'étudier le processus d'analyse statistique de traces de contact. Tout d'abord, nous avons résumé les hypothèses pré-analyse, en se basant sur des travaux antérieurs. En utilisant trois jeux de données de la littérature, nous avons illustré leur in uence sur les paramètres des distributions de probabilité estimées. Nous avons montré que les contacts de durée nulle et l'estimation de la borne inférieure de la loi de Pareto ont un fort impact, tandis que la longueur de trace, la définition d'inter-contact ou les nœuds externes jouent un rôle moins important. Nous nous sommes ensuite tournés vers l'étude du délai et du taux de délivrance obtenus avec les traces filtrées, en considérant cette fois uniquement les contacts de 0 seconde, les noeuds externes et la longueur de trace. Nous constatons que les observations faites pour l'analyse statistique ne s'appliquent pas nécessairement aussi aux performances, et dépendent clairement du jeu de données retenu.

Considérant que des modèles précis doivent être dérivés de données réalistes, et que nous avons précédemment montré que les hypothèses préliminaires peuvent fortement affecter ces dérivations, le champ d'utilisation de ces modèles s'en trouve réduit: comme un modèle donné ne ferait que capturer une situation précise, il serait impropre à la

généralisation. Cela représente une autre limite des traces de contact. En fait, ces conclusions dépendent fortement de la nature des ensembles de données, et peuvent ne pas s'appliquer partout. Ceci motive la check-list proposée ici. Cette liste n'est certainement pas exhaustive, et devrait être étendue, au moins avec les autres hypothèses mentionnées dans cette section. Nous souhaitons aussi étendre notre travail aux métriques par paires pour comparaison avec les distributions agrégées. Il serait également intéressant d'effectuer une analyse de performances réseau, mais cette fois sur des traces synthétiques tirées des distributions statistiques, et de comparer avec les résultats obtenus sur les traces initiales. Enfin, étendre l'analyse de performances à d'autres hypothèses initiales ou d'autres métriques pourrait aussi être utile.

3.4 Analyse de traces réelles : de la collecte à la mise à l'échelle

Les traces réelles sont un outil très prisé des chercheurs en DTN souhaitant apporter une touche de réalisme à leurs travaux. Trois cas d'utilisation peuvent être dégagés : analyse de liens sociaux, évaluation de performances réseau ou analyse statistique. Dans cette partie, nous avons identifié les différents facteurs pouvant affecter les traces, pendant la collecte, le filtrage et la mise à l'échelle, en se basant sur 4 jeux de données existants. Ceci nous permet ensuite de proposer des recommandations pour l'usage des traces dans les trois cas d'étude mentionnés précédemment. Cette étude peut être vue comme le prolongement de celle entamée pour les traitements statistiques au chapitre 5, présentée dans la section précédente.

Les traces sont en effet un matériau de prédilection pour l'étude des DTNs, qu'elles soient issues d'expériences réelles ou de modèles synthétiques. Cependant, les traces réelles sont affectées par plusieurs facteurs, tels que la taille du réseau étudié, la période d'échantillonnage ou encore la technologie radio utilisée. Dans cette section, nous nous intéressons au cycle de vie complet des traces réelles, de leur capture initiale jusqu'à l'usage qui en est fait ensuite en passant par d'éventuelles opérations de filtrage.

Pour cela, nous avons sélectionné 4 traces existantes (Section 6.2). Trois d'entre elles (Infocom 2005, MIT, Rollernet) ont déjà été utilisées précédemment dans nos travaux, alors que la quatrième (Humanet) représente une expérience plus récente pour laquelle les auteurs ont beaucoup soigné la procédure de collecte. Ces 4 traces présentent une variété intéressante, que ce soit au niveau des longueurs d'enregistrement, dates de capture, nombre de nœuds ou environnements capturés.

Trois cas d'utilisation peuvent être dégagés de la littérature existante (Section 6.2.2). Premièrement, il est possible de faire une analyse sociale, pour identifier les communautés ou des liens sociaux. On peut aussi se servir des traces comme d'une validation plus réaliste de nouveaux protocoles, en complément d'un essai sur trace synthétique, comme cela est souvent le cas dans la littérature. Enfin, l'analyse statistique est généralement employée pour changer l'échelle des traces initiales, comme nous le reverrons dans la suite.

Au moment de la collecte de traces (Section 6.3), le but est de parvenir à obtenir

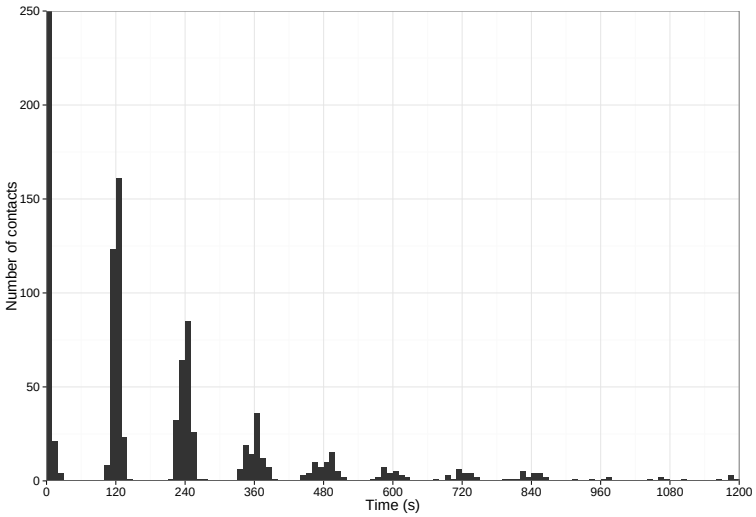


Figure 3.4: In uence de la période d'échantillonnage sur les durées de contacts enregistrées.

un enregistrement d'une situation réelle. Il est cependant nécessaire dès cet instant de prendre en compte de nombreux paramètres. Ainsi, le système d'exploitation et le matériel employé pour la collecte imposent des limites sur le processus de collecte et les paramètres accessibles. La synchronisation des horloges des différents nœuds ou le format de stockage final des données obtenues sont aussi des questions importantes. Mais l'un des facteurs les plus importants à ce stade est sans doute la période d'échantillonnage. Trop longue, celle-ci manque des opportunités de contact ; trop courte, elle épuise prématurément les batteries. Mais elle in uence également les données collectées, dans la mesure où tous les contacts enregistrés auront des durées multiples de la période d'échantillonnage, avec généralement un pic à 0 seconde. Nous montrons l'existence de cette structure particulière sur un exemple de trace réelle dans la figure 3.4.

Une fois la trace collectée et publiée, d'autres chercheurs souhaiteront généralement s'en servir pour tester leurs propositions. Là encore, de nombreux facteurs doivent être pris en compte (Section 6.4). On peut notamment penser à la définition des intercontacts, ou au choix de mettre ou non une symétrie des contacts. Nous montrons sur un exemple en Figure 3.5 que cette symétrie peut avoir des conséquences importantes, par exemple sur le calcul du degré. Selon l'expérience à mener, il peut aussi être souhaitable de supprimer une partie des données, soit des nœuds immobiles ou extérieurs qui pourraient polluer la trace, ou des paires rarement en contact pour lesquelles une analyse statistique serait faussée par manque de données. Deux autres filtres peuvent cependant avoir une in uence plus grande. Il est en effet courant dans les traces de fusionner deux contacts séparés d'un seul sondage sans contact, car on suppose que la perte de visibilité résulte d'un problème radio et pas d'une mobilité réelle. Ce filtrage a cependant l'inconvénient de créer des contacts artificiellement longs, pouvant surestimer ensuite la capacité réelle d'écoulement du trafic. Ceci est vrai pour des traces réelles comme pour des traces

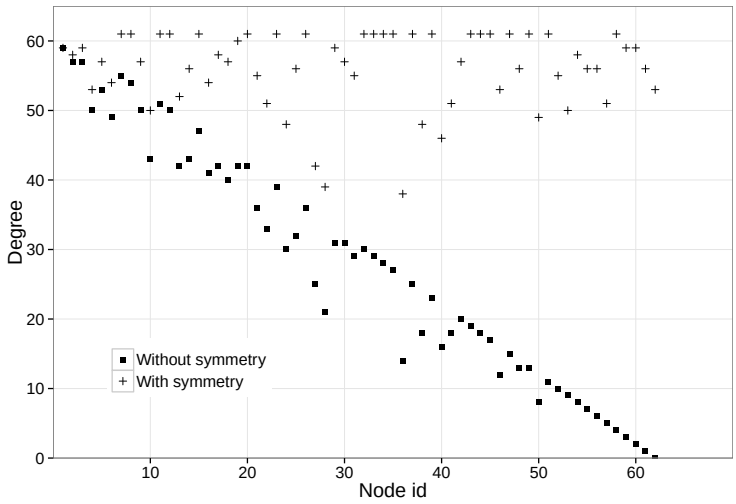


Figure 3.5: In uence de la prise en compte de la symétrie dans le calcul du degré de la trace symétrique Rollernet.

synthétiques. Il est aussi monnaie courante de supprimer les contacts trop longs ou trop courts, dont en particulier les très nombreux contacts de durée nulle déjà mentionnés auparavant. Il faut cependant noter que cette durée nulle est en fait un artefact du processus d'échantillonnage. Le problème qui se pose lorsqu'on supprime de tels contacts, par exemple pour pouvoir appliquer des outils statistiques utilisant des logarithmes, est l'in uence sur les intercontacts. En effet, la cohérence voudrait qu'on fusionne deux valeurs si celles-ci ne sont plus séparées par un contact.

Même après les différents filtrages exposés ci-dessus, d'autres traitements peuvent encore s'avérer nécessaires. Par exemple, la trace peut être trop longue ou comporter trop de nœuds, ou encore des périodes particulières (pauses, nuits, vacances...) qu'on peut souhaiter dans certains cas exclure de l'analyse. A l'opposé, la trace peut ne pas être assez longue ou assez dense : c'est pour remédier à ce problème que deux solutions de mise à l'échelle, CTG et EMO, ont été proposées dans la littérature. Nous les présentons en Section 6.5. Dans les deux cas, l'approche envisagée est la même : extraire de la trace réelle des lois statistiques supposées capturer la caractéristique du réseau, pour ensuite les réutiliser et obtenir une trace synthétique à l'échelle souhaitée présentant les mêmes caractéristiques. Le focus est mis sur les contacts et intercontacts, agrégés et/ou par paire (pas d'analyse sociale ou de mobilité). Ces approches présentent cependant de nombreux inconvénients. Par exemple, elles n'utilisent qu'une seule distribution statistique pour toutes les paires, là où la littérature suggère qu'il en faudrait plusieurs. Nous avons montré que ce système perdait en fait une partie de l'information : les traces synthétiques obtenues présentent une longueur différente et manquent les valeurs d'intercontact les plus longues, qui bien que rares re ètent une structure caractéristique (dans notre exemple, la présence de nuits dans la trace originale). De plus, il existe différentes définitions des outils et lois statistiques utilisées. La borne minimale présente

pour certaines distributions et pas d'autres rend aussi la comparaison difficile. Enfin, la validation pose également problème. Celle-ci est faite avec des distributions statistiques agrégées, qui offrent certes un moyen simple et pratique (une seule courbe à comparer) pour la validation, mais ont l'inconvénient d'être aussi des entrées des algorithmes. En mélangeant toutes les paires, elles masquent également certaines différences qui ont pourtant une in uence sur les performances finales du réseau, comme cela a été démontré dans la littérature.

Cette étude nous permet à présent d'établir des recommandations pour les futurs utilisateurs de traces réelles (Section 6.6). Pour la production, il est important de choisir un matériel et logiciel proche de celui envisagé pour l'application, et une période d'échantillonnage faisant un compromis entre précision et consommation électrique. Notons cependant que depuis les premières collectes de traces en 2004, les attentes des utilisateurs en matière d'autonomie ont évolué, ce qui peut potentiellement permettre d'augmenter la précision. Il est important à ce stade de filtrer les données le moins possible. Concernant le filtrage, nous proposons des recommandations pour les trois cas d'utilisation définis en introduction. Si le filtre sur la longueur, la définition de l'intercontact ou la symétrie peuvent s'appliquer si nécessaire pour tous les cas, les autres filtres nécessitent davantage de précautions. Il faut aussi noter que les filtres ne sont pas indépendants : par exemple, changer la longueur de trace réduit le nombre d'événements, tout comme la suppression des valeurs les plus courtes. Pour la mise à l'échelle, nous ne voyons pas d'objection à se limiter à un sous-ensemble des données initiales. En revanche, le passage à une échelle plus grande en termes de nœuds et/ou de temps total nous semble déconseillé, en raison des nombreuses limitations des approches actuelles énoncées plus haut. Si de nouveaux outils, prenant en compte les différences entre les paires et utilisant de nouvelles approches de validation, venaient à être développés, le passage à l'échelle pourrait selon nous être alors envisagé.

Par souci d'exhaustivité, nous listons ensuite en Section 6.7 les diverses questions ouvertes et ré exions distillées au fil du chapitre. En particulier, nous discutons du choix des hypothèses (valeurs extrêmes, définition de l'ICT, période d'échantillonnage) de filtrage ou de collecte. Puis nous ouvrons sur les problèmes liés à l'analyse statistique, comme les paires supprimées ou non modélisées, les méthodes de comparaison entre distributions ou la validation des approches de passage à l'échelle.

En conclusion, les traces de contact collectées en conditions réelles représentent un moyen simple et usuel d'ajouter un peu de réalisme dans des travaux de recherche. Pour l'étude des DTNs, trois cas d'utilisation peuvent être identifiés : analyse sociale, évaluation de performances et analyse statistique. Dans ce travail, nous avons montré que la représentation de la réalité donnée par les traces est en fait affectée par de très nombreux facteurs. Nous avons aussi listé des filtres appliqués dans des travaux existants, et proposé des recommandations sur leur usage. Concernant les approches de mise à l'échelle des traces, nous montrons que là encore plusieurs limitations entrent en jeu. En raison du nombre important de facteurs à considérer pour l'usage des traces dans l'évaluation des performances, nous sommes arrivés à la conclusion que les modèles synthétiques devraient être regardés avec plus d'attention. En effet, ceux-ci offrent un contrôle total sur

leurs paramètres. Évaluer leur réalisme reste cependant un défi ouvert, dans la mesure où ceci est le plus souvent fait par comparaison avec des traces réelles.

Dans cette première partie du manuscrit, nous nous sommes donc intéressés à deux types d'entrées possibles pour notre système d'émulation. Nous avons développé et validé un modèle analytique de prédiction du taux de pertes dans un réseau DTN constitué de sources, de destinations et de relais, dans le cas de mémoires limitées à un seul paquet. Les cas d'utilisation très limités de ce modèle nous poussent ensuite à nous tourner vers les traces de contact réelles. Inspirés par des approches existantes de mise à l'échelle, nous montrons qu'en réalité les filtrages effectués avant l'analyse statistique des traces sont loin d'être anodins, et que ces conclusions ne s'étendent pas nécessairement à l'étude de performances. Ensuite, nous étendons cette analyse de filtrages à l'ensemble du cycle de vie des traces, de la collecte jusqu'à la mise à l'échelle. Ceci nous permet de formuler des recommandations pour de futures études.

3.5 HINT : un émulateur DTN pour le développement d'applications

Dans cette section, nous passons à présent à la conception de l'émulateur proprement dit. Bien que l'émulation ne soit pas un sujet nouveau, il convient de noter qu'aucune des approches existantes de la littérature pour l'évaluation de performances DTN ne s'attache particulièrement au développement d'applications. Pourtant, la complexité de tels réseaux oblige les développeurs à s'intéresser à la caractérisation du réseau. Malheureusement, ceux-ci ne sont pas toujours correctement outillés pour utiliser efficacement cette information.

Nous choisissons ici de nous focaliser sur les contacts et intercontacts pour la couche de lien réseau, ce qui nous permet de nous abstraire des considérations de mobilité. A cet effet, deux propriétés nous semblent nécessaires : des connexions bi-directionnelles, et la possibilité pour un nœud donné d'être simultanément en contact avec plusieurs autres nœuds. Pour réellement aider les développeurs, nous devons fournir un outil s'intégrant facilement dans le processus de développement. Nous définissons pour cela plusieurs critères à remplir. L'émulation doit se faire en temps réel. Nous nous basons aussi sur les contacts, afin d'abstraire de façon simple la mobilité des nœuds. Pour ajuster les paramètres des algorithmes ou de l'application, notre émulateur devra supporter la modification de paramètres en temps réel, ainsi qu'un monitoring en temps réel également pour en observer les effets sans attendre la fin de l'expérimentation. L'émulateur devra aussi être totalement transparent pour l'application testée, offrir une certaine répétabilité des expériences (condition nécessaire pour le débogage) et être facilement disponible. Ce dernier point signifie notamment une possibilité d'installation sur des ressources matérielles plus limitées que celles d'un *testbed* traditionnel.

Nous présentons ensuite l'architecture de notre émulateur nommé HINT en Section 7.3. A un haut niveau d'abstraction, nous utilisons deux types de nœuds. Les nœuds réels sont des téléphones mobiles sous Android qui exécutent l'application à

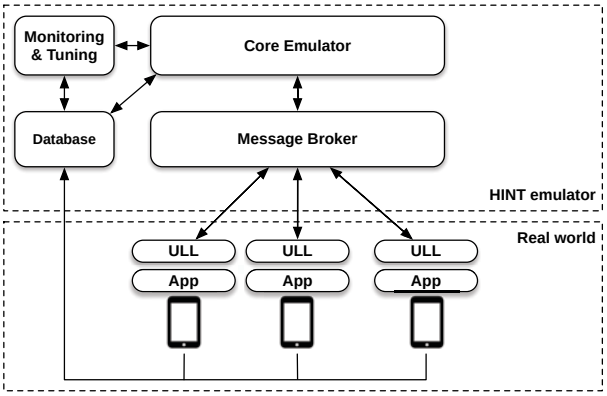


Figure 3.6: Schéma des 5 briques constituant HINT.

tester, et connectés en Wi-Fi à l'émulateur. Les nœuds virtuels sont quant à eux gérés par l'émulateur, tout comme les connexions entre nœuds. Tous ces nœuds, réels comme virtuels, interagissent dans un même réseau émulé. Pour l'émulateur proprement dit, nous proposons une architecture en 5 parties distinctes, comme visible sur la Figure 3.6. Le cœur d'émulateur a pour mission de définir et exécuter le scénario choisi, en générant notamment les différents événements nécessaires, et en les exécutant en temps réel au moment voulu. Les décisions de routage et la gestion mémoire sont aussi sous sa responsabilité. Un *message broker* est ensuite utilisé pour gérer ces événements et les relayer aux différents nœuds, et permet de représenter de manière originale les mémoires des noeuds du réseau, réels comme virtuels. Une couche de lien réseau abstrait la communication entre l'application réelle et l'émulateur, en offrant un service permettant d'insérer n'importe quelle application dans l'environnement d'émulation. Une base de données est aussi utilisée. Celle-ci sert à sauvegarder les paramètres du scénario, l'état des noeuds ainsi que certaines des données nécessaires au calcul de performances. Enfin, un module permet la collecte et l'affichage de statistiques réseau en temps réel via une interface Web, ainsi que l'édition de certains paramètres. Nous montrons ensuite que cette architecture répond effectivement aux spécifications exprimées précédemment, et qu'elle peut bel et bien s'exécuter sur des ressources matérielles beaucoup plus restreintes que celles d'un *testbed*.

Pour mettre en oeuvre cette architecture, nous choisissons d'utiliser le langage Python, avec MongoDB comme base de données et RabbitMQ comme *message broker*. Dans le coeur d'émulateur, un ordonnanceur permet de classer et de déclencher les différents événements au moment voulu. Concernant les contacts et intercontacts, il est possible d'utiliser des entrées provenant de distributions statistiques ou de traces réelles selon l'expérience envisagée. Au total, HINT supporte 6 algorithmes de gestion de mémoire et 4 algorithmes de routage, ainsi que l'expiration par TTL. Au sein du *message broker*, chaque noeud possède 3 queues : Inbox pour la réception, Outbox pour l'envoi de messages et Storage pour les messages dont le noeud est destinataire. L'interface Web de suivi propose 3 niveaux de lecture (réseau, paire, noeud) et offre la plupart des métriques

DTN usuelles telles que le délai, l'occupation mémoire ou la distribution des temps de contact. La couche de lien réseau, quant à elle, est développée sous forme de service Android à intégrer dans l'application à tester.

3.6 Validation et utilisation de l'émulateur

Il est à présent temps de mettre en pratique l'implémentation de l'architecture proposée au chapitre 7. Cette mise en pratique est réalisée en deux temps : une validation, pour vérifier que les spécifications sont tenues et que les résultats fournis font sens par rapport aux approches existantes, puis l'application à une étude de cas.

Dans un premier temps, nous validons donc HINT (Section 8.1). Une première expérience consiste à vérifier que l'émulateur est effectivement capable de passer à l'échelle en termes de nombre d'événements (contacts et intercontacts) exécutés par seconde. Le critère employé ici est la capacité à atteindre un délai d'exécution restant borné au cours de l'expérimentation. Nous appliquons donc une charge allant de 1 à 1225 événements par seconde sur une durée de 20 secondes, et mesurons le retard d'exécution sur 5 itérations. Les résultats se trouvent en Figure 3.7. La comparaison avec les valeurs obtenues en analysant des traces réelles montre que HINT peut gérer un nombre d'événements par seconde (300, avec un seuil de 1 s) supérieur à la valeur maximale de 146 enregistrée sur la trace Rollernet, pourtant très dense. Ceci valide le passage à l'échelle pour les événements.

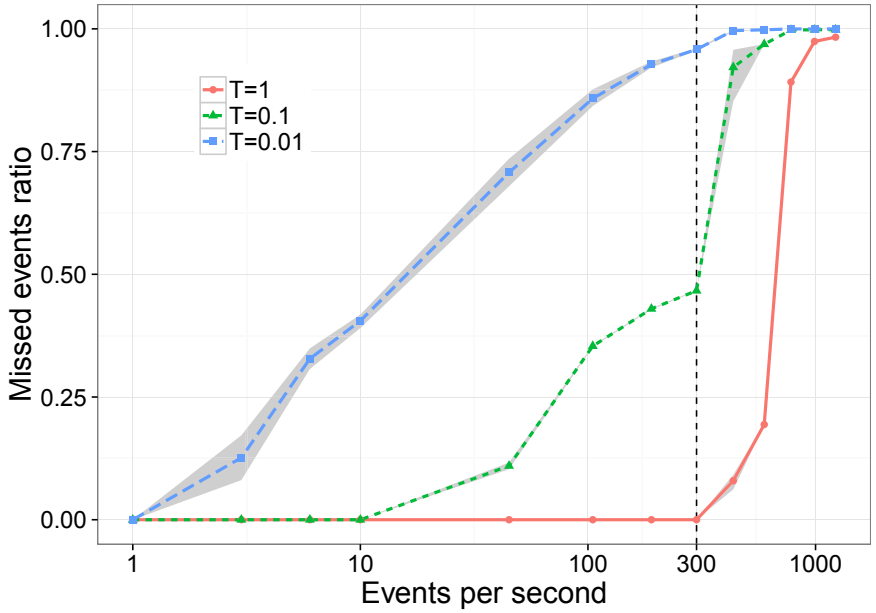


Figure 3.7: Pourcentage d'événements retardés pour différents seuils (T en secondes) et charges système (événements/s) (échelle log-linéaire).

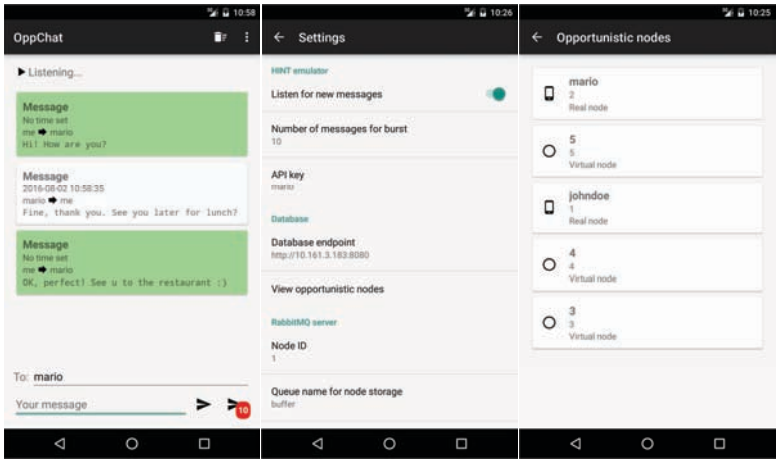


Figure 3.8: Captures d'écran d'illustration de l'application OppChat.

Dans une seconde expérience, nous vérifions le réalisme de notre émulateur, en comparant les résultats obtenus avec ceux issus d'un simulateur (ONE) à hypothèses identiques. Il ressort de cette comparaison que sur cette expérience les résultats fournis par HINT, ici en termes de délai et de taux de délivrance, sont plus pessimistes que ceux issus de ONE. Nous pensons que la source de ces différences provient d'une différence d'implémentation dans la gestion des contacts : HINT ne démarre de transferts qu'au début d'un contact, là où ONE se rafraîchit à chaque pas de temps de simulation.

Dans un second temps (Section 8.2), nous utilisons l'émulateur HINT dans le cadre de la mise au point d'une application Android de messagerie opportuniste dénommée OppChat, visible en Figure 3.8. Plus précisément, nous analysons deux situations différentes : une discussion entre deux personnes à travers un réseau opportuniste d'une part, et un contingent de noeuds envoyant de manière périodique leur statut à un noeud unique d'autre part. A chaque fois, nous analysons le délai et taux de délivrance obtenus au cours de l'expérimentation. Dans les deux cas, le taux de délivrance obtenu en fin d'expérimentation est très proche de 1. Nous constatons cependant dans le cas de la discussion entre utilisateurs qu'au vu des délais constatés (médiane à 72 secondes), avoir une véritable discussion avec un message toutes les 15 secondes semble impossible.

Nous mentionnons ensuite d'autres pistes pour de futures expériences, comme par exemple tester d'autres applications ou se comparer avec des modèles analytiques tels que celui présenté au chapitre 4. Enfin, nous présentons de possibles développements supplémentaires du système.

3.7 Conclusion et perspectives

Dans cette thèse, nous nous sommes intéressés à l'évaluation de la performance des réseaux DTN. Bien que plusieurs approches existent déjà, aucune ne répond réellement aux besoins des développeurs d'applications. Le besoin d'intégrer des noeuds réels et

des applications nous a conduits à considérer l'émulation.

Dans un premier temps, nous nous sommes intéressés à la production d'entrées pour un tel système. Nous avons proposé un modèle analytique pour le taux de pertes, mais comme tout modèle analytique, les hypothèses sous-jacentes limitent fortement son champ d'application. Nous nous sommes donc ensuite tournés vers les traces réelles. Nous avons listé les différentes hypothèses prises lors de l'analyse statistique de traces, et montré leur in uence sur les résultats obtenus en termes de distributions statistiques et de paramètres. Si dans certains cas l'impact est limité, il convient malgré tout d'être attentif à ces hypothèses. Une étude complémentaire sur les performances réseau obtenues a montré que les conclusions faites sur les distributions statistiques ne s'appliquent pas nécessairement aux performances.

Dans un second temps, nous avons exploité les conclusions de la première partie pour proposer HINT, un émulateur léger de réseau opportuniste destiné aux développeurs d'applications. Nous avons tout d'abord listé des spécifications pour notre système, utilisées ensuite pour en déduire une architecture d'émulation. Cette architecture a ensuite été implémentée en Python, avec l'aide d'une base de données MongoDB et d'un *message broker* RabbitMQ. L'étape suivante est la mise en oeuvre du système obtenu. Pour cela, nous avons procédé à la validation de HINT, en évaluant sa capacité à tenir un nombre d'événements par seconde suffisant, et à fournir des résultats en accord avec ceux obtenus avec d'autres approches. Enfin, nous avons mis à profit HINT pour évaluer les performances d'une application de *chat* opportuniste DTN OppChat dans deux cas particuliers d'utilisation : une discussion entre deux utilisateurs à travers un réseau opportuniste d'une part, et un envoi régulier de messages de statut à un noeud unique d'autre part.

Nous avons finalement présenté quelques perspectives pour poursuivre les travaux en Section 9.1. Dans la même optique que les modèles analytiques de performance, nous avons mentionné la recherche d'une "fonction d'émulation" capable d'abstraire le réseau. Plusieurs pistes d'amélioration pour HINT ont aussi été évoquées, tout comme les approches de mise à l'échelle de traces réelles introduites au chapitre 6. Enfin, nous nous sommes interrogés sur la représentativité des analyses statistiques et sur leur capacité à réellement capturer les caractéristiques du réseau étudié. Une ouverture à d'autres disciplines, en particulier les sciences humaines pour les aspects réseaux sociaux, a été brièvement mentionnée.

Part I

Emulation Inputs: Models and Traces

Chapter 4

A DTN model: drop ratio in one-packet bu er networks

Contents

Among the approaches used for DTN performance assessment presented in Chapter 2, analytical models have the benefit of simplicity of use (at the expense of costly initial derivations) and very low setup cost. In this chapter, we introduce a new mathematical model for the study of an underrepresented metric in the literature, namely the drop ratio. We also show its interest for dimensioning the kind of network assumed here, a sensor network made of sources and destinations.

Note: part of this work already appeared in IEEE AOC 2015, under the title *A Markov chain model for drop ratio on one-packet bu ers DTNs.* http://dx.doi.org/10.1109/WoWMoM.2015.7158190

4.1 Introduction

This chapter studies the drop ratio for messages in DTNs with finite buffers. Until now, most of the efforts to characterize DTNs have focused on delay and delivery ratio. These works usually assume there is sufficient buffer space to route messages by assuming infinite buffers, as seen in Section 2.2. In this work, we inverse the problem to focus on the most constrained scenario: nodes with buffer capacity for just one message. We propose a Continuous-Time Markov Chain (CTMC) to model the drop ratio. To the best of our knowledge, there is no previous work with such approach. Our model is introduced for a network with homogeneous ICTs and a simple extension for a two-class heterogeneous ICT case. In both cases, the modeled drop ratio fits simulation results obtained with synthetic traces.

A state of the art on existing analytical models has already been provided in Chapter 2. It can be seen from it that while almost all papers consider the delay, only a few studied the drop ratio, which is the motivation for this chapter. More precisely, we want to find the maximum number of sources that the network can accept before experiencing losses due to buffer over ow at the relaying nodes. This can be seen in some way as the capacity of the network.

In this chapter, we will consider a sensor network, with nodes performing measurements and relaying the data among peers until it reaches a collection point. Consequently, a group of nodes acts as sources and another one acts as destinations, without any overlap between the two. The remaining nodes only act as relays. We also consider that destinations are undistinguishable from each other: indeed, a message only needs to reach any of the collection points, instead of a specific one or all of them.

Nodes can carry only one message. Although quite restrictive, this use case can be seen as a situation where nodes equipped with bigger buffers end up having only one free space in their buffer. This is a kind of limit before they start dropping messages due to buffer saturation.

4.2 Drop model

In this section, the basic notation and hypotheses of the models are defined. We then introduce the two models: homogeneous exponential intercontact time distribution and a restricted heterogeneous case with two classes of nodes, each class having a different parameter for the exponential law. Limitations and suggested extensions will be discussed later in Section 4.4.

4.2.1 Notation

We consider a DTN with N identical nodes $\mathcal{N} = \{1, 2, 3, \quad , N\}$, each one with a buffer capacity of one message. Among these nodes, $S < N$ are sources and there are M initial copies of the same message in the network. Notice that $S = M$ due to the buffer restriction to one single message. The M messages are delivered to any of the $D < N - S$

destinations. Unless stated differently, we consider $D = 1$. Recall that there is no overlap between sources and destinations. Intermediary nodes act as simple forwarders of those M messages. Hence, no extra copies are created in the evolution of the process, and no other messages will be generated (no background traffic). The goal is to determine the distribution of dropped messages and the drop ratio over time.

Let $0 \leq t_{i,j}(1) \leq t_{i,j}(2) <$ be the successive encounter times among nodes i and j. We consider that the transmission time of a message is negligible with respect to the time it takes to two nodes to meet one another, and consequently assume instantaneous contacts. It follows that the n-th intercontact time between i and j is $ict_{i,j}(n) = t_{i,j}(n+1) - t_{i,j}(n)$. Since each node can keep only one message, each time a contact occurs we try to forward it. Hence, when node i and j meet, either: (i) only one of the nodes has a message, and it is instantaneously transmitted, or (ii) both nodes have a message, then one is chosen at random to instantaneously transmit its message while the other drops its message. There is no other way to drop a message: no interference is considered, messages never expire (no TTL considered), and all contacts are successful.

4.2.2 The (d, c) model: homogeneous case

For the homogeneous case, let us assume that the processes $\{t_{i,j}(k),\ k \quad 1, \quad i = j \in \mathcal{N}\}$ are mutually independent and homogeneous Poisson processes with rate $\lambda > 0$. Hence the random variables $\{ict_{i,j}(k),\ k \quad 1, \quad i = j \in \mathcal{N}\}$ are mutually independent and exponentially distributed with mean 1 λ.

To calculate the number of dropped messages, we introduce a continuous time Markov chain $(X(t), t > 0)$. The states of the chain are (d, c), where d represents the number of dropped messages and c the remaining number of message copies. At the beginning, there are only the M message copies in the network; hence, our initial state is $(0, M)$. The transitions from a state (d, c) are either to $(d, c-1)$ when a message is delivered or to $(d+1, c-1)$ when a drop occurs. The rate to encounter a destination will be $D\lambda$. Since we have c nodes with a copy of the message, the former transition will happen with rate $cD\lambda$. The latter occurs with rate $\frac{c(c-1)}{2}\lambda$ given the number of combinations where two nodes with a message meet. In Figure 4.1, we detail the possible transitions for the general case. The absorbing states of the chain are in the form $(d, 0)$ with $0 \leq d \leq M-1$. Notice that when reaching the absorbing states, we must impose some boundary conditions. Since the last message will be delivered with probability 1 (because it cannot be dropped under the assumptions of our study), a transition from state $(d, 1)$ to $(d+1, 0)$ is impossible. Figure 4.2 presents the complete Markov chain for M.

We can easily calculate the probabilities of going out from the (d, c) state. Indeed, the embedded Markov chain for $X(t)$ allows to write the probabilities to jump between

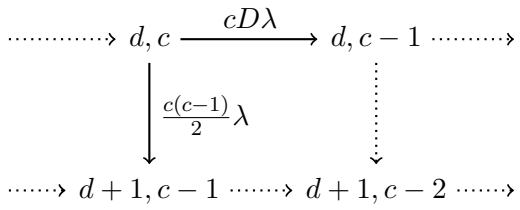


Figure 4.1: Chain transitions from a given state (d, c). Either a destination is found among D possibilities, leading to state $(d, c-1)$, or there is a drop, leading to $(d+1, c-1)$. State $(d+1, c-2)$ is included to show the progression of states.

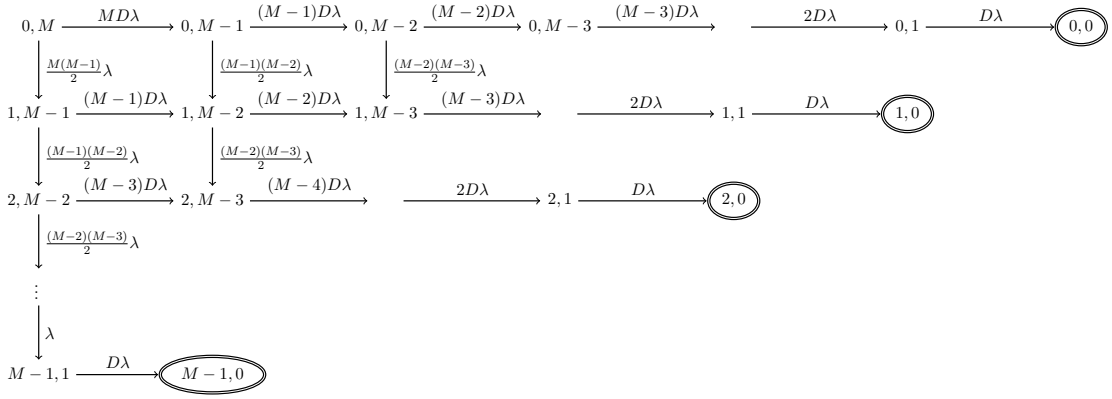


Figure 4.2: Complete Markov chain for the drop process. When there are no more messages to deliver, we reach a final state $(d, 0)$, with d being the number of messages dropped.

states as shown in Equation (4.1).

$$\begin{aligned} P((d,c) \quad (d,c-1)) &= \frac{cD\lambda}{cD\lambda + \frac{c(c-1)}{2}\lambda} = \frac{2D}{c+2D-1} \\ P((d,c) \quad (d+1,c-1)) &= \frac{\frac{c(c-1)}{2}\lambda}{cD\lambda + \frac{c(c-1)}{2}\lambda} = \frac{c-1}{c+2D-1} \end{aligned} \tag{4.1a}$$

It is important to notice that in this case, the probabilities are independent of the process arrival rate λ. Therefore, for any $\{t_{i,j}\}$ defined as before, the same drop ratio results can be expected. The absorbing states probabilities are defined as:

$$P_M(d) = P(X_\infty = (d, 0)\ X(0) = (0, M)),\ 0 \leq d \leq M-1 \tag{4.2}$$

Since the process is a feed-forward process, these probabilities can be easily calculated with a dynamic programming algorithm. The drop rate distribution is defined as the expected value of reaching the absorbing states over the number of starting messages:

$$(M) = \frac{1}{M} \sum_{d=0}^{M-1} d\, P_M(d) \tag{4.3}$$

4.2.3 The (d, c_1, c_2) model: heterogeneity with two classes

We now extend the (d, c) model to an heterogeneous case. Here, the N nodes are split into two classes, $\mathcal{C}_1$ and $\mathcal{C}_2$, with $\mathcal{C}_1 = N_1$ and $\mathcal{C}_2 = N_2$ such that $N = N_1 + N_2 + D$. For convenience, we impose $D = 1$. For simplicity, the destination does not belong to any class of nodes. There are $M = M_1 + M_2$ messages distributed as M_1 in $\mathcal{C}_1$ and M_2 in $\mathcal{C}_2$.

The heterogeneity is defined such that the processes $\{t_{i,j}\}$ have different rates for each class: independent homogeneous Poisson processes with rate λ_1 in class $\mathcal{C}_1$ and with rate λ_2 in class $\mathcal{C}_2$, while inter-class interactions are given by independent and homogeneous Poisson processes with rate λ. As before, it follows that the random variables $ict_{i,j}$ are mutually independent and exponentially distributed with the reciprocal of λ_1, λ_2 or λ according to the case. They consequently define different speeds for the group encounter rates (and hence the waiting times between connections). For instance, this allows to model one group that will have a faster delivery than the other. This means more nodes with free buffers, and fewer drops. For the symmetry of the problem, the destination meets nodes from $\mathcal{C}_1$ with rate λ_1 and nodes from $\mathcal{C}_2$ with rate λ_2.

Building upon the previous subsection, the states are modeled as (d, c_1, c_2) where d is the number of drops, c_1 is the number of copies in class $\mathcal{C}_1$ and c_2 is the number of copies in class $\mathcal{C}_2$. We identify three different kinds of transitions from the state (d, c_1, c_2):

1. Delivery transitions: we meet the destination with rate λ_i from class $\mathcal{C}_i$. Because there are c_i message copies, the transition rates are $c_1\lambda_1$ for $(d, c_1 - 1, c_2)$ and $c_2\lambda_2$ for $(d, c_1, c_2 - 1)$;

2. Drop transitions: the number of combinations in class $\mathcal{C}_i$ where two nodes with a message meet is $\frac{c_i(c_i-1)}{2}\lambda_i$. Also we count the c_1c_2 combinations where two nodes from different classes having a message meet with rate λ 2[1]. It follows that the transition rates are $c_1(c_1 - 1)\lambda_1 \ 2 + c_1c_2\lambda \ 2$ for $(d + 1, c_1 - 1, c_2)$ and $c_2(c_2 - 1)\lambda_2 \ 2 + c_1c_2\lambda \ 2$ for $(d + 1, c_1, c_2 - 1)$;

3. Inter-class transitions: in this case we count the number of combinations where a node from class $\mathcal{C}_i$ passes a message to a free node from class $\mathcal{C}_{j\neq i}$. This happens $c_i(N_j - c_j)$ times with rate λ. It follows that the transition rates are $c_1(N_2 - c_2)\lambda$ for $(d, c_1 - 1, c_2 + 1)$ and $c_2(N_1 - c_1)\lambda$ for $(d, c_1 + 1, c_2 - 1)$.

Figure 4.3 presents the general transitions from a state (d, c_1, c_2) to all possible states described before. Like in the previous subsection, there are M absorbing states in the

[1]We can split the inter-class Poisson processes into two counting processes with rate p and $(1-p)$. The splitting probability corresponds to the drop probability when two nodes have a message. Since we take one at random, we have $p = 1 - p = 1 \ 2$.

form $(d, 0, 0)$ with $0 \leq d \leq M-1$. Notice that we arrive to the absorbing state either from $(d, 1, 0)$ with rate λ_1 or from $(d, 0, 1)$ with rate λ_2. Again, we need to rule out some transitions. For instance, from (d, N_1, c_2) we cannot go to $(d, N_1 + 1, c_2 - 1)$ (there are only N_1 nodes in C_1) or from $(d, 0, c_2)$ to $(d, -1, c_2 + 1)$, because the number of copies or drops cannot be negative. Due to its size, the complete chain is not presented here.

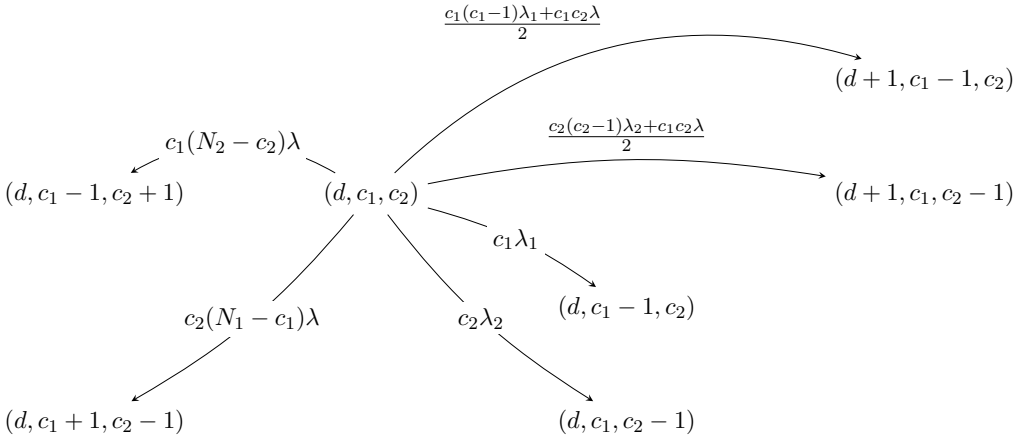


Figure 4.3: Chain transitions from a given state (d, c_1, c_2). We observe the increase in number of states and transitions w.r.t the (d, c) model shown in Figures 4.1 and 4.2: inter-class transitions $(d, c_1 - 1, c_2 + 1)$ and $(d, c_1 + 1, c_2 - 1)$; delivery transitions $(d, c_1 - 1, c_2)$ and $(d, c_1, c_2 - 1)$; and drop transitions $(d + 1, c_1 - 1, c_2)$ and $(d + 1, c_1, c_2 - 1)$.

To calculate the drop probabilities, we use the fact that the chain previously defined is an absorbing Markov chain. The steps are as follows:

1. enumerate the N_s valid states of the chain (d, c_1, c_2) with $0 \leq d \leq M-1$, $0 \leq c_1 \leq N_1$ and $0 \leq c_2 \leq N_2$. This allows to define the mapping from $1 \leq i \leq N_s$ to state $s_i = (d', c_1', c_2')$ for all the valid states in the chain.

2. define the matrix A with coefficients a_{ij} as the rate transform from state s_i to state s_j and the matrix R with coefficients r_{ij} as the rate from state s_i to absorbing state s_j^*. Notice that the size of matrix A is $N_s \quad N_s$ and the matrix R is $M \quad N_s$.

3. finally, we define $B = (I - A)^{-1}R$ where the i^{th} row is the distribution of absorbing states if the initial state is s_i.

Since the absorbing states correspond to drop states, it follows that B is the dropping distribution $P_M(d)$.

Some insights on the modeling for the general heterogeneous case are presented in Section 4.4.

4.3 Simulation setup

In this section, we present the main results of the comparison between a simulated DTN network and the modeled CTMC results. We generate pairwise ICTs distributed as defined in Section 4.2, with the same number of events for each pair (100). We then run an event-driven simulation to perform the forwarding of the messages in the network, observe drops and calculate the drop ratio for a given configuration. We compare the simulated and predicted drop ratio for both (d, c) and (d, c_1, c_2) models. Table 4.1 presents an overview of the simulations covered in this section. The theoretical model results have been obtained with MATLAB, while the continuous time simulations are implemented with R [67].

All simulations are run with $N = 100$ nodes and buffer size $B = 1$ message. We repeat each simulation 10 times and provide the average results within a 95% confidence level. The network occupancy is defined as $\pi = S\ N$. The number of sources is gradually increased to represent the following percentages $\pi \in \{1, 2, 5, 10, 15, \quad , 80, 95, 98\}$.

Table 4.1: Summary of simulations and their parameters (π: network occupancy, D: destinations, : increasing).

	π	D	Model	Parameters
E1		1	(d, c)	$\lambda = 500$
E2			(d, c)	$\lambda = 500$
E3		1	(d, c_1, c_2)	$\lambda_1 = 2\ 5,\ \lambda_2 = 200,\ \lambda = 66\ 6$
E4		1	(d, c_1, c_2)	$\lambda'_1 = 10,\ \lambda'_2 = 100,\ \lambda' = 55$

4.3.1 Homogeneous case: single destination

As defined in Table 4.1, we now present the results of simulation $E1$ with $\lambda = 500$. All pairs are assumed to follow the homogeneous case with $\{t_{i,j} \quad PoissonProcess(\lambda)\}$ and $\{ict_{i,j} \quad \exp(\lambda^{-1})\}$.

Figure 4.4 shows how increasing the number of sources (hence the network occupancy) increases the drop ratio, as predicted by the (d, c) model. In this figure we plot the drop ratio for each repetition (red points). We also plot the average interpolation up to a 95% confidence interval envelope (gray area within the error bars). Again, we see how close the simulated and model results are. Indeed, the maximum difference between the average case for the 10 repetitions and the model is 0 04. We will see a bigger difference if the variance is included (points dispersion), especially with a small number of sources: with less sources, the probability of dropping a message is lower, but not zero (bigger variance), whereas with more sources, chances of eventually dropping are almost 100% (smaller variance).

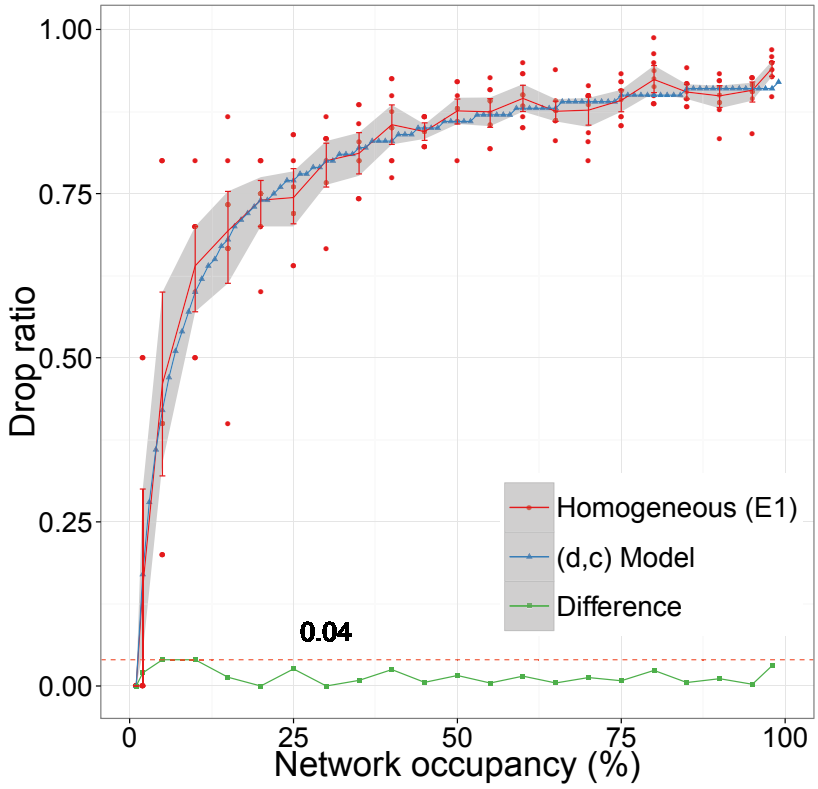


Figure 4.4: Results for the homogeneous case ($E1$): we see how close (d, c) model-predicted values and simulated values are.

4.3.2 Homogeneous case: anycast

Simulation $E2$ is an extension of $E1$, but with an increasing D. We set $\lambda = 500$ and increase both the number of sources and destinations, keeping the total node count to 100. Recall that no node can simultaneously be a source and a destination. Since all destinations are indistinguishable, each time we meet one a message is delivered. Figure 4.5 shows the evolution of the drop ratio when increasing the number of destinations for different values of S such that $\pi \in \{5, 10, 20, 30, 40, 50, 60, 70\}$. The number of destinations D varies up to the maximum amount of free nodes on each case, i.e. $1 \leq D < N - S$. We can see how adding destinations reduces the drop ratio because of the increased delivery probability. Again, the model matches the simulated results well. As in the $E1$ case, the variance is lower with a higher number of sources.

This graph allows to define how many source nodes can be put in an anycast sensor network with a given number of destinations to keep the drop ratio bounded. It can also be used the other way round, to derive the number of destinations required to achieve a target drop ratio with a given number of sources. For example, with 20 anycast

destinations, a drop ratio lower than 25% can be achieved for any network occupancy below 40%.

4.3.3 Heterogeneity with two classes

In this section, we discuss both simulations $E3$ and $E4$ with the main results for the two-class (d, c_1, c_2) model. There are still $N = 100$ nodes, with $D = 1$ destination and the rest of nodes divided in two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of the same size ($N_1 = 50$ and $N_2 = 49$ respectively). We increase the number of sources selecting randomly M_1 nodes from $\mathcal{C}_1$ and M_2 nodes from $\mathcal{C}_1$ so that $M = M_1 + M_2$. We define $\lambda_1 = 2\ 5$, $\lambda_2 = 200$, $\lambda = 66\ 6$ for $E3$ and $\lambda'_1 = 10$, $\lambda'_2 = 100$, $\lambda' = \frac{\lambda'_1+\lambda'_2}{2} = 55$ for $E4$.

Figure 4.6a presents the compared model results for three cases: (d, c) model and (d, c_1, c_2) for $E3$ and $E4$ simulations. First, we notice that the homogeneous case performs worse than both two-class heterogeneous cases. Recall that the probabilities of the (d, c) model are independent of λ, while in the (d, c_1, c_2) model they are not. Indeed, we see how the assigned values for the $E3$ case give a lower drop ratio than $E4$. This is due to the encounter rates defined: on simulation $E3$, we have $\lambda_1 < \lambda < \lambda_2$. This means that nodes in class $\mathcal{C}_1$ meet more frequently than the nodes in $\mathcal{C}_2$, and nodes in $\mathcal{C}_2$ meet more frequently a node from $\mathcal{C}_1$ than one from their own class.

Consequently, messages in the first class will reach the destination more frequently than the ones in the second class, and messages in the second class can also benefit from the transfer opportunities to the first class to be in turn forwarded to the destination. Since messages are being delivered more frequently, the drop ratio decreases in comparison with the homogeneous case. On contrast, on simulation $E4$ the drop ratio increases. Thinking in terms of interactions between nodes, the total number of pairs is 4950, with 1225 pairs per group. The number of interactions is distributed as $\mathcal{C}_1$ 25%, $\mathcal{C}_2$ 25% and 50% for the inter-class interactions. In $E4$, the rates are in the same configuration as $E3$, but with a difference in the order of magnitude as $\lambda'_1 < \lambda' < \lambda'_2$: values λ'_1 and λ'_2 in $E4$ are closer to a unique λ value than λ_1 and λ_2 in $E3$. Hence, more than half of interactions behave similarly in terms of encounter frequency, which explains why $E4$ is closer to the homogeneous case than $E3$.

Figure 4.6 shows how the two-class model matches our simulations in both number of drops and drop ratio. In this figure we include the model for both $E3$ and $E4$, as well as the simulation results in a 95% confidence interval. Like in the homogeneous case, fewer sources means a bigger variance on the simulated results.

4.4 Discussion

In this section, we discuss the limits of the models presented in this chapter. We also introduce two possible extensions: a larger buffer or a completely heterogeneous case.

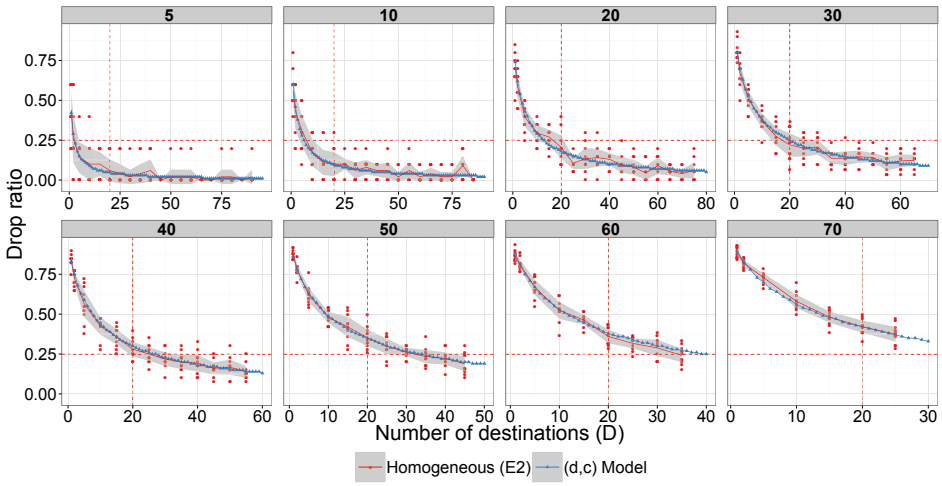
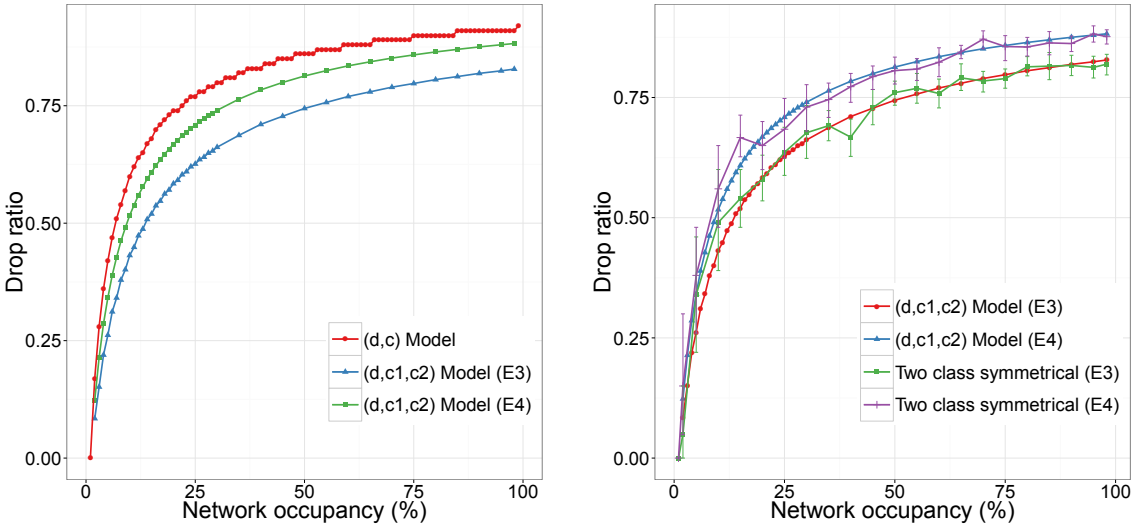


Figure 4.5: Results for $E2$: each graph represents a fixed network occupancy π. As predicted by the model, adding destinations decreases the drop ratio. The number of destinations needed to get a 25% drop ratio can be seen for each case.



(a) Comparison between the (d, c) and the (d, c_1, c_2) models. For the (d, c_1, c_2) model, different ICT parameters are considered ($E3$ and $E4$).

(b) Results for $E3$ and $E4$: we see how close the (d, c_1, c_2) model-predicted values and the simulated values are for the symmetrical case.

Figure 4.6: Drop ratio for two-class simulations $E3$ and $E4$.

Table 4.2: The possible (d, c) model extensions: larger buffer and/or heterogeneity.

	Homogeneous	Heterogeneous
$B = 1$	(d, c)	$(d, c_1, c_2, \quad , c_N)$
$B = B > 1$	$(d, b_1, b_2, \quad , b_B)$	$(d, c_1, c_2, \quad , c_N)$

4.4.1 Model limits

In our model, we impose that the node pairs follow $\{t_{i,j} \quad PoissonProcess(\lambda)\}$ and $\{ict_{i,j} \quad \exp(1 \ \lambda)\}$. This use of pairwise exponentially distributed intercontact times is one of the limits. Indeed, the type of mobility model behind this kind of intercontact times is usually random (random waypoint, random walk, etc) [34], whereas human movement follows other distributions.

Then, computational limits arise, related to the capacity to represent all the states and transitions of the Markov chain in a computer. Going from the (d, c) model to the (d, c_1, c_2) model increases the number of states and transitions, making it a harder problem to deal with. Indeed, we observed some important issues when implementing our algorithms. A naive implementation is likely to use all the memory of the system trying to represent the states and transitions. For the (d, c_1, c_2) model, the definition of the matrices must be done using sparse matrices (since there will be a lot of zero values). The storage of the states and its easiness to perform a search will also impact the running time (for instance a linear search in a vector of states versus a binary search). We see that the (d, c) model is orders of magnitude faster than performing the corresponding simulations. However, we find equivalent times when running the (d, c_1, c_2) model and simulations.

4.4.2 Model extensions

As seen in Section 4.3, the (d, c) and (d, c_1, c_2) models introduced here allow us to derive the drop ratio when increasing the number of sources in the network. The simplicity of the model is linked to the strictness of our hypotheses. One could say that studying the forwarding of one message in the network is not pertinent. However, we see this approach as a starting point to study future extensions to our model. Indeed, Table 4.2 shows possible extensions to other cases: (i) a single-packet buffer heterogeneous case (ii) larger ($B > 1$) buffer homogeneous case (iii) larger buffer heterogeneous case. We now discuss the main ideas under these extensions.

Single-packet bu er, heterogeneous case

The states of the CTMC are defined as $(d, c_1, \quad , c_N)$, where d is the number of dropped messages and c_i is a binary variable to represent if the node i has a message or not. The number of states grows as $\mathcal{O}(2^N)$.

Larger buffer, homogeneous case

The states of the CTMC can be defined as $(d, b_1, \ldots, b_B)$, where d is the number of dropped messages and b_i the number of buffers with i messages ($0 < i \leq B$). In this case, the number of states grows as $\mathcal{O}(2^B)$.

Larger buffer, heterogeneous case

Finally, we define the states of the CTMC as $(d, c_1, \ldots, c_N)$, where d is the number of dropped messages and c_i is an integer variable to represent the number of copies that the node i has in its buffer ($0 \leq c_i \leq B$). The number of states grows as $\mathcal{O}(2^N)$.

We can see extensions of the model are possible, but the number of states required in such cases grows exponentially. The number of transitions also explodes, making it much more difficult to provide and calculate the CTMC and its embedded Markov chain.

4.5 Conclusion

In this chapter, we introduced a model for the drop ratio in a DTN network where buffers are limited to one packet. We applied it to a network defined as a set of sources which emit one message (sensors), destinations to receive them (collection points), and intermediary relay nodes. Simple forwarding is applied to the messages, to avoid any replication which would increase the number of drops. Two network types were considered here, each one corresponding to a different model: the (d, c) model for a fully homogeneous network (same ICT distribution for all pairs of nodes), and the (d, c_1, c_2) two-class heterogeneous case (same distribution but different parameters). Both models were then validated against simulations. The homogeneous model was considered with an increasing number of destinations, while the heterogeneous one was considered with different values for the node encounter frequency. We showed how the choice of the values modified the message drop performance. The interest of the proposal for network dimensioning in terms of number of sources and/or destinations to achieve a given drop ratio was also demonstrated. Possible model extensions include a fully heterogeneous case, dealing with buffers larger than one message and testing the two-class model with a broader range of encounter frequencies.

At this point, it is noteworthy that the ability to derive the results, thanks to the use of Markov chain formalism, relies on several simplifying assumptions. One of the most common in the literature, which also appears in this work, is the exponential character of pairwise intercontact times (along with instantaneous and always successful contacts). Should one of these assumptions be relaxed, the whole model would have to be rewritten almost from scratch. This would be a huge problem, given the time needed for this and the fact that a study would typically explore a large parameter space. For example, it may be necessary to try several routing protocols in order to compare their performance. The realism of some assumptions can also be questioned: for example, studies of pairwise distributions on real datasets [23, 87] concluded that the best distribution was log-normal

instead of exponential. However, using the log-normal distribution would prevent the use of Markovian theory.

Another issue of such models is the number of metrics which can be derived from them. Here, each state of the Markov chain corresponds to a number of drops and a number of copies left in the network. This makes our proposed model only suitable to the study of drops (and indirectly, the delivery ratio: in this model, messages are either dropped or received). Other possible metrics like the delay cannot be provided by it. This is problematic for our emulation use case, since a developer will typically wish to reproduce all impairments happening in the network. Restricting the study to a subset of them, or considering each impairment separately, will not be sufficient for proper evaluation. Hence, if one is also interested in other metrics (delay, for example), another model would have to be derived or used, if it already exists in the literature. In an application development context, deriving a new analytical performance model cannot be envisioned due to the time and skills required. On the contrary, simulators like the ONE [47] can provide as many metrics as needed, if the necessary report classes exist. If they do not, creating them is quite straightforward. Simulators are however dependent on the inputs they are given: if an unrealistic mobility model is being used, then performance results derived from it are also likely to be unrealistic.

A possible way to overcome this limitation could be the use of contact datasets collected in real situations. At first glance, one may expect from such datasets that they give a proper picture of the reality. Indeed, several papers have been using them as a way to prove applicability of new proposals, such as routing protocols, not only to synthetic mobility models, but also to real conditions. They seem therefore to be good candidates for inputs to an emulation system, where being as close to the reality as possible is key. Contact traces are consequently the focus of the next two chapters.

Chapter 5

Real datasets statistical analysis

Contents

We now turn to real traces analysis. Indeed, real traces are a popular material among practitioners, because it allows both experiment reproducibility and the inclusion of realistic characteristics. It is however possible not to use them directly, but to concentrate instead on data *derived* from these traces, such as statistical distributions. For the present work, which focuses on DTN network emulation, this use case cannot be neglected at first glance. Obtaining these distributions and studying the factors which could affect this derivation is the main topic of this chapter.

Note: part of this work has been presented at the ACM CHANTS 2014 workshop, under the title *Revisiting pitfalls of DTN datasets statistical analysis*. http:

//dx.doi.org/10.1145/2645672.2645683

Contact traces collected in real situations represent a popular material to assess the performance of a DTN. These traces usually require some preprocessing to be fully usable. More specifically, several assumptions can be made prior to performing the statistical analysis of contact and inter-contact times. In this chapter, we first classify these assumptions, and analyze their effect on the statistical characterization of three well-known datasets. We also identify some pitfalls in dataset analysis that might strongly impact on the conclusion made by the experimenter. Based on our own experience, we subsequently propose a preliminary checklist to help researchers avoid undesired ambiguities or misunderstandings in further studies. We finally complete our study by assessing the impact of the assumptions on performance results obtained with the datasets.

5.1 Introduction

With the advent of ubiquitous use of smartphones and other mobile devices, opportunistic networks and DTNs have gained a lot of attention. The challenge of analyzing or forecasting their performance largely relies on datasets of CT and ICT between nodes. A common practice is to generate synthetic data from analytical models. To overcome their inherent lack of realism, large datasets have been built from live measurement collections [5, 28, 76]. Because such measurements are very hard to perform, only few research teams have managed to provide such datasets. They have the advantage of capturing a part of reality, and researchers have repeatedly used them to give a much more realistic avor to their input data. However, some datasets are used much more often than others, forming a kind of *de facto* standards. Several studies have also been made in order to derive statistical models from these datasets for CT and ICT distributions, either pairwise [23, 87] or aggregated across all pairs of nodes present in the data [46, 61].

To cope with the difficulty of collecting new datasets, the authors of EMO [83] propose a novel approach based on a simulation tool using encounter events between nodes, which takes as input CT and ICT distributions derived from existing datasets. Thanks to the use of statistical distributions instead of raw traces, this tool allows scaling of the network both in terms of time span and number of nodes. A similar approach is also presented in [16]. However, the realism achieved with datasets is questionable. As an illustration, a recent study shows that contact-based simulations do not take into account the limitations on node buffers and available transfer bandwidth [73]. Hence, these simulations do not provide trustworthy estimations of delivery ratios and delays in DTNs.

In this chapter, we attempt to warn experimenters that the statistical handling of these datasets may alter their nature in a somewhat hidden manner. We focus on the statistical analysis of contact datasets and extract from previous literature addressing this topic a set of hypotheses and choices made before the analysis (Section 5.2). Then, we classify them with respect to their in uence on the distribution fittings obtained for

three widely used datasets (Section 5.3). This allows us to propose a dataset statistical analysis checklist to avoid hidden pitfalls, following our own experience (Section 5.4). To complete this study, Section 5.5 illustrates the impact of some of the assumptions, but this time on performance results obtained from traces. We conclude and give directions for future work in Section 5.6.

5.2 Background

Contact traces have been widely used in the literature on DTNs as a starting point for modeling and predicting performance. Consequently, this section presents the traces which will be exploited in this chapter, along with the tools and assumptions used for statistical analyses.

5.2.1 Datasets used

We have used three datasets for our study, namely Rollernet, MIT Reality Mining and Infocom 2005. We will not go into further details, because the characteristics of these datasets were already presented in Chapter 2.3. The reason for using these datasets is that they are among the oldest data made available (in 2006, 2004 and 2005 respectively), and consequently among the most used in the literature. Hence, they can be seen as being part of the *de facto* standards mentioned previously.

Some limitations of the use of contact traces for simulations have already been pointed out in the literature. In [91], the wireless technology used (e.g., Bluetooth) was shown to miss many contact opportunities, thus in uencing subsequent protocol performance studies. In [73], the authors also emphasize the need to take the real data transfer capacity of each contact opportunity into account. This includes contact durations, available bandwidth or node buffer occupancy, parameters which are not recorded by contact traces in the datasets.

5.2.2 Statistical tting tools

Among the papers considered, several approaches are used to fit the data to well-known statistical distributions. The simplest one is to plot the data with an adequate scale according to the model tested, and perform a graphical fitting. With this method, Hui et al. [43] for instance fit the aggregated ICT in the Infocom 2005 dataset to a Pareto law. The authors of [46] subsequently apply a similar approach to the same type of data (including the Infocom 2005 dataset), finding this time a Pareto law but with an exponential decay.

Other authors prefer to use statistical tools for their fitting studies: the work in [83] relies on the KS test, while the authors of [23] (and later [87] or [61]) choose the Cramer-von Mises test. Unlike the Cramer-von Mises test, which can operate on discrete data, the KS test is restricted to continuous values and distributions. Considering that the times recorded in the traces and expressed in seconds are in fact discretized samples of a continuous variable, we choose to use the KS test for our studies.

5.2.3 Assumptions used in previous analyses

In this section, we discuss the choices and assumptions a typical practitioner has to make prior to performing a statistical analysis of such contact traces. Based on previous articles, they are as follows:

Choice of nodes: it is possible to analyze only the contacts between nodes participating in the experiment (often called internal) or to include also other external nodes which were observed despite not being actively participating in the data collection. Although Hui et al. [43] consider both in their analysis, most studies are typically restricted to internal nodes only. Indeed, these external nodes may introduce a subtle bias in the conclusions due to their nature. We can also mention that the trace preprocessing script of the Adyton simulator [60] performs a removal of external nodes as its first step.

Pair symmetry: if a contact is recorded between node i and node j, a contact between j and i does not necessarily exist at the same time. This is especially true when Bluetooth is used to produce the trace. At the very best, and according to our observations, it may be possible to observe a short contact in one direction during a much longer one in the opposite direction. While some papers, such as [87], explicitly assume symmetry, most of the time asymmetrical pairs are considered;

Minimum number of contacts: when studying pairwise metrics, a lower bound on the number of contacts is generally chosen to ensure that the statistical analysis is made on a sufficient number of samples. For example, a threshold of 9 contacts is considered in [87], while the authors of [23] use 4 contacts. Interestingly, no arguments are provided to justify the precise choice of values;

0-second contacts: the traces considered for this study all exhibit a large percentage (at least 40%) of contacts recorded with a duration of 0 second. Yet, some papers such as [83] appear to discard all contacts shorter than the measurement granularity, thus removing a substantial part of the data. On the contrary, the authors of [87] choose to include these 0-second contacts in their analysis by extending them to 1 second;

Time span: one might be tempted to use only a portion of a dataset, especially when the original experiment duration is long. Another justification can be to avoid specific time periods which may exhibit different characteristics. In the case of the MIT dataset, which lasts for 284 days, only 180 days[1] are considered in [83], while the authors of [61] and [30] use 246 days;

Intercontact de nition: while it is common practice to define the inter-contact as the interval of time when two nodes are not in contact, the difference between the

[1] According to our ndings, this corresponds to the 284 days with all weekends, MIT holidays and public holidays removed.

beginning of two successive contacts is instead used in [83]. For clarity, we show both definitions in Figure 5.1;

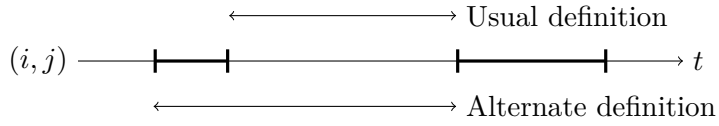


Figure 5.1: Illustration of the two definitions of intercontact times for a given node pair (i, j). Bold areas denote contact periods.

Power-law parameters: almost all articles proposing a statistical fit consider the Pareto distribution, as it has been shown in [43] to characterize the aggregated ICT and CT Complementary Cumulative Distribution Functions (CCDFs) of the Infocom 2005 dataset. In this case, the CCDF can be written:

$$F(x) = \left(\frac{x_{\min}}{x}\right)^{\alpha-1}, \tag{5.1}$$

for x $x_{\min} > 0$, > 1. $x_{\min}$ is the lower bound, and the exponent. The lower bound $x_{\min}$ is arbitrarily set as the measurement granularity in [83], [20] and [19]. In [22], the authors propose a framework for the estimation of *both* parameters (and $x_{\min}$) from the experiment data, aiming to replace error-prone graphical estimation techniques for the exponent.

5.3 Impact of initial assumptions on dataset analyses

We now present the consequences of the previous assumptions on the results of the statistical analysis. Unless otherwise stated, the baseline assumptions for all datasets used in this section consider asymmetrical pairs among internal nodes, with the usual inter-contact definition, namely the time difference between the end of a contact and the beginning of the next one. No minimum number of contacts is enforced for pairs, and the measurement granularity is used as the lower bound $x_{\min}$ for the Pareto law. Finally, 0-second contacts are extended to 1 second, so that no inter-contacts are modified or removed. In the following subsections, we take each assumption separately while keeping the other parameters unchanged.

We chose to focus on the *aggregated* contact times, although our study also covered aggregated inter-contact times. We tried to fit them to three distributions: exponential, log-normal and power law (Pareto), which are the most represented in the literature. Note that it would be possible to adapt the code to deal with other distributions, as long as the corresponding parameter estimation techniques can be found in existing libraries or implemented.

All analyses were carried out using the R statistical software [67]. We implemented ML estimators to obtain the parameters of all the distributions considered, and used the

companion code of [22] for the Pareto distribution functions and parameter estimation. Since traces record integer time values, inter-contact and contact times typically exhibit ties (i.e., some numerical values appear more than once). Consequently, in order to use the correct empirical Cumulative Distribution Function (CDF) in the KS test, we reimplemented this test with the aid of the R `ecdf` function. Due to the presence of logarithms in the estimation formulas, 0-second contacts had to be extended to 1 second when taken into account, following [87].

For completeness, the formulas used for ML parameter estimation are as follows:

Exponential distribution:

$$\lambda = \frac{1}{\frac{1}{n}\sum_{k=1}^{n} x_k} \tag{5.2}$$

Lognormal distribution:

$$= \frac{1}{n}\sum_{k=1}^{n} \ln x_k \tag{5.3}$$

and

$$= \sqrt{\frac{\sum_{k=1}^{n} (\ln x_k - \quad)^2}{n}} \tag{5.4}$$

Finally, the exponent estimation for the usual Pareto distribution is

$$= 1 + n \left(\sum_{k=1}^{n} \ln \frac{x_k}{x_{\min}} \right)^{-1} \tag{5.5}$$

No estimation formula is provided for $x_{\min}$, this value being either arbitrarily chosen, or found in an empirical manner (using Clauset algorithm [22]).

We will present the effect of 0-second contacts, Pareto estimation methods, trace length, external nodes and ICT definitions on the curves and the resulting fittings. The other hypotheses identified in Section 5.2.3 are not covered here because they are not applicable to aggregated values (symmetry has no in uence on them, and a minimum number of samples is not necessary because aggregation operates on enough data); they are consequently left for future work.

5.3.1 0-second contacts

First, we study the in uence of 0-second contacts. In this subsection, three possibilities are investigated:

removal and merging of the surrounding inter-contacts;

extension to the arbitrary value of 1 second (baseline) without ICT modification;

removal of all contacts and inter-contacts shorter than the measurement granularity.

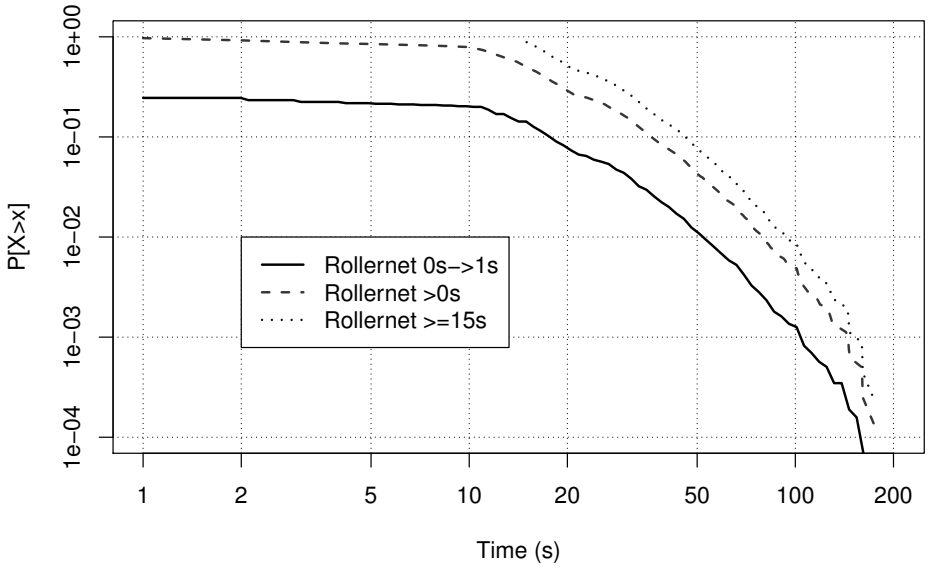


Figure 5.2: In uence of the treatment of 0-second contacts on aggregated contact time distribution for the Rollernet dataset (log-log scale).

The dataset used here for illustration consists of the 5000 first seconds of Rollernet, as in [87]. It should be noted that this dataset assumes symmetrical pairs. The results for each hypothesis are presented in Figure 5.2 for the aggregated contact time distribution.

As can be seen from Figure 5.2, the three curves exhibit large variations. This removal of the shortest contacts also strongly impacts the fitted parameters. This is due to the fact that they represent an important part of the trace (75% for 0-second contacts in this case). For the Rollernet case, the best fits appear to be Pareto laws for all three hypotheses considered. Even when the laws do not change, parameters remain of utmost importance, as their value can largely condition the behavior of the network [46].

We now show the results for these same hypotheses, but on the 284-day version of the MIT trace. For clarity, we did not represent the curve corresponding to the removal of all contacts shorter than the measurement granularity, as it would overlap the curve obtained with contacts strictly longer than 0 second. The resulting curves can be found in Figure 5.3.

As in the previous case, taking into account 0-second contacts creates major changes on the curves. This time however, we also notice a change of distribution for the best fit: we move from Pareto (= 1 534 and $x_{\min} = 300$) when all contacts are considered to log-normal (= 7 562, = 1 130) when 0-second contacts are discarded.

When using traces to infer network capacity, 0-second contacts are paradoxical: on one hand, they last 0 second, so no data can be transmitted (unless data exchange is considered instantaneous, in which case only inter-contact times need to be modeled);

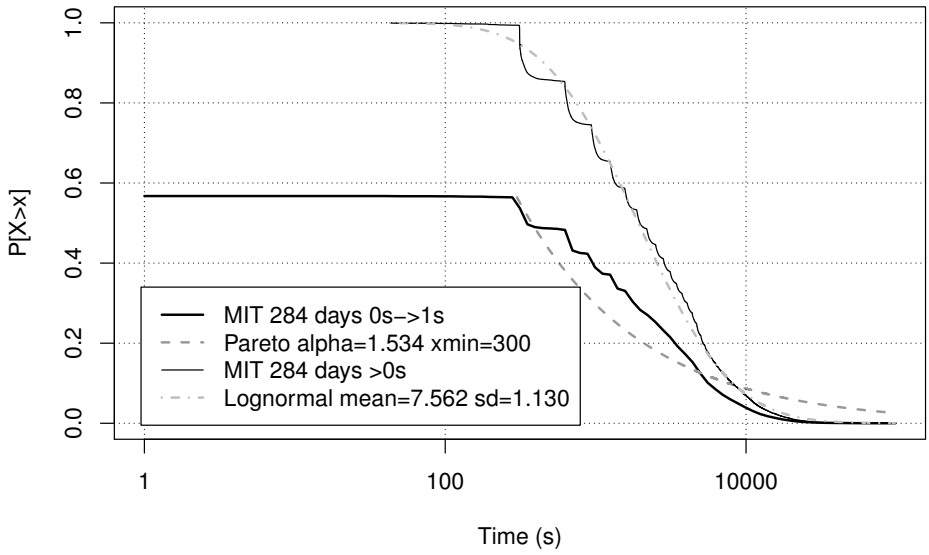


Figure 5.3: In uence of the treatment of 0-second contacts on aggregated contact time distribution and associated fittings for the MIT dataset (log-linear scale).

on the other hand, they do appear in the trace, which implies that some data was indeed exchanged between the nodes during the experiment. In order to solve this paradox, we believe that the choice made in [87], namely to extend the contacts to 1 second, is the best trade-off between data transmission opportunity and short contact time. However, we just showed that this choice could have a strong impact on the results of statistical fittings. It is therefore crucial to check that the treatment of 0-second contacts is consistent with the use cases of resulting models. More details on this apparent paradox will be given in Chapter 6.

In this subsection, we only focused on a lower bound for contact time. It is also possible to discard the longest contacts, or even to filter out both the shortest and the longest ones, as we will see later in Chapter 6. The variability of the results exposed here, although restricted to a study on lower bounds, clearly calls for a clearer and universal choice of filtering methods, a call which was already made in [23].

5.3.2 Pareto lower bound estimation

We now focus solely on the Pareto law. In Section 5.2.3, two different methods for the lower bound estimation were presented: one consisting of simply setting it to the measurement granularity, whereas the other one requires the use of the algorithm introduced by [22]. This algorithm selects the lower bound $x_{\min}$ and the associated exponent as the ones which provide the best fit with the data, i.e., the smallest distance D in the KS test.

We compare the output of these two methods for the aggregated CTs in the Infocom 2005 [76] dataset. The results are shown in Figure 5.4. In this example, the distances D are 0 199 when the lower bound is set to the measurement granularity, i.e., $x_{\min} = 120$; and 0 027 when the lower bound is estimated as $x_{\min} = 1402$.

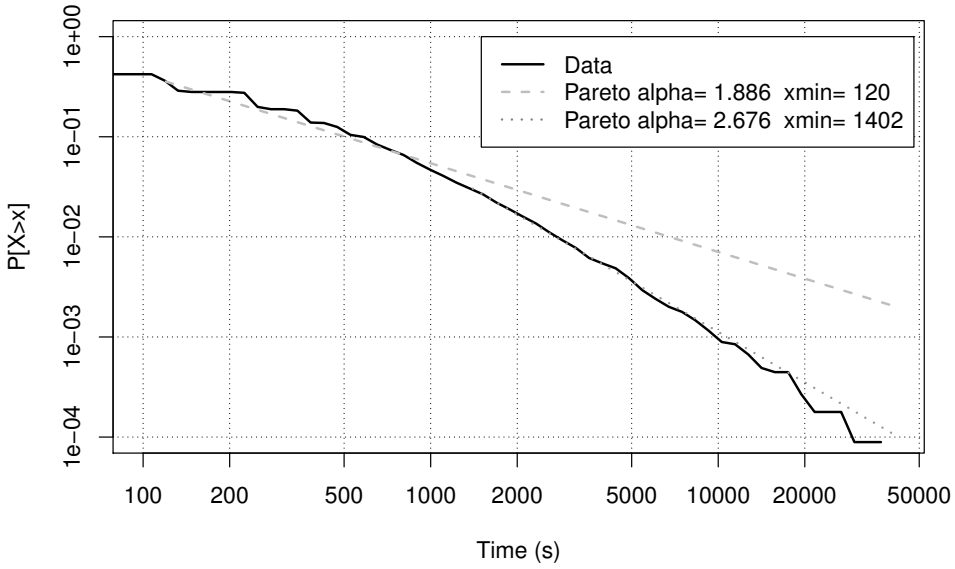


Figure 5.4: Pareto fittings of the aggregated contact time distribution for the Infocom 2005 dataset (log-log scale).

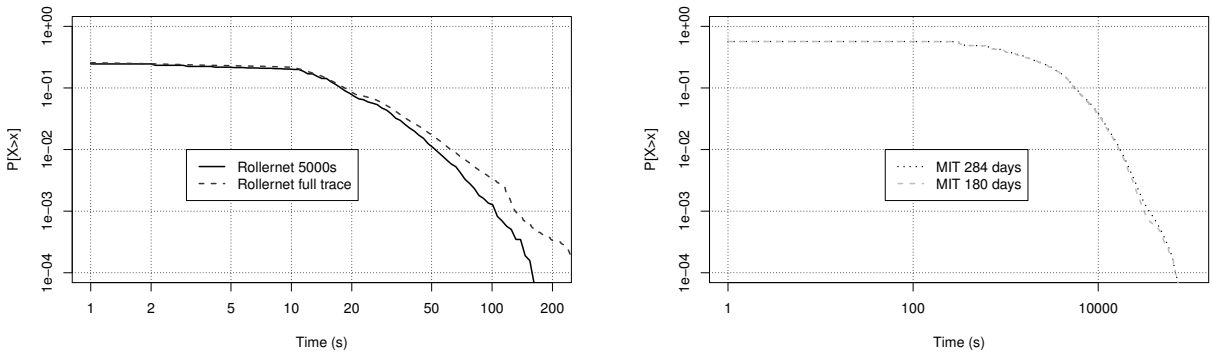
As can be seen from Figure 5.4, there are high discrepancies between the parameters provided by both approaches. We also found similar differences while applying the same treatments to the other datasets considered here, for contacts as well as inter-contacts.

In spite of being very contrasted, the results are not contradictory. Instead, they simply show that the lower bound giving the best fit with respect to the KS test distance D metric is not necessarily the measurement granularity, and is typically higher.

For modeling purposes, being able to fit the sole tail of the data is problematic, as it amounts to discarding most of the trace. In this case, estimating the lower bound $x_{\min}$ would only take 3% of the total data into account. When arbitrarily setting the lower bound to the measurement granularity, this percentage rises to almost 36%, which still leaves a large part of the data without a model. To deal with this issue, some authors (for example in [83]) choose to use the measurement granularity as their lower bound and to discard all contacts shorter than this threshold.

5.3.3 Trace length

Datasets are not always directly usable as a whole. For example, there may be periods when some of the experiment nodes were not functioning properly [28] or the conditions



(a) Rollernet

(b) MIT

Figure 5.5: In uence of trace length on the aggregated contact time distribution for two datasets (log-log scale).

of the experiment were different (such as a break during a rollerskating tour [87], or holidays between school terms [28]). Since these time periods may not exhibit the same properties as the rest of the dataset, it could be desirable to remove them before the analysis. Hence, we analyze the in uence of truncation on the empirical distributions, using two datasets which have already been studied in truncated versions: MIT and Rollernet. More precisely, we consider the MIT dataset in full (284 days) and without weekends and holidays (180 days) as in [83], while Rollernet will be used either in full, or only for the first part before the break (5000 seconds).

The results are presented in Figure 5.5 for the aggregated contact times. For both examples considered, the change in duration only has a small impact on the curves. We also found similar results with the aggregated inter-contact times. The case of the MIT dataset is particularly interesting: while the 180 days correspond to only 63% of the total trace length, they concentrate 90% of the contacts recorded. This difference can be explained by the fact that the experiment was conducted among students, who are more likely to meet each other while on campus than during weekends or holidays. Due to the low variation in the total number of contacts between the two versions, similar results can be expected in both cases.

In the Rollernet [5] case, the two parts of the dataset are much better balanced than in the MIT case. First, there is no discrepancy between the time percentage and the proportion of contacts included in this percentage. Furthermore, the 5000 first seconds and the rest of the dataset were collected in similar conditions, namely a rollerskating tour, except for the intermediate 20-minute break when participants were likely to be less mobile. For these reasons, we can also expect only minor changes to the curves and the fitted parameters.

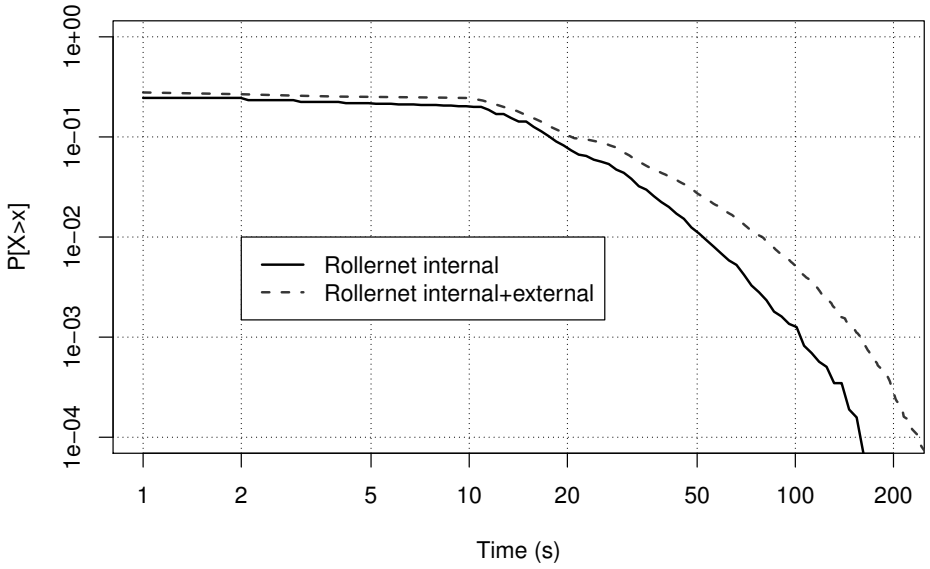


Figure 5.6: In uence of external nodes on the aggregated contact time distribution for the Rollernet dataset (log-log scale).

5.3.4 External nodes

Traces are not restricted to contacts occurring among the experimental devices; hence, one can choose whether to use or not these external contacts in his study. However, as external devices may follow different mobility patterns, it may be interesting to evaluate the impact of their use on the empirical distributions. We perform this evaluation for the 5000 first seconds of the Rollernet dataset. The results are shown in Figure 5.6.

We find out that taking into account external nodes for the statistical analysis of the Rollernet trace has a limited in uence on the curves. This is true for both contact and intercontact times empirical distributions. In [43], external nodes were also found to behave similarly as internal nodes in the Infocom 2005 case. Our analysis of this same dataset leads to the same conclusions. These findings are of great interest, as the environments in which the datasets were collected are rather different. For Infocom, external nodes are most likely to be other participants at the conference, while in the Rollernet case they are either participants to the tour or passersby with no particular link with the tour. If all attendants to a conference may be expected to have similar mobility patterns, regardless of their participation in the trace collection, passersby will typically be slower and less mobile than roller-skaters. The number of external devices recorded is also very different between the two traces: 182 for Infocom and 1050 for Rollernet, much like the associated number of contacts: 5757 for Infocom and 43076 for Rollernet. With such discrepancies regarding the percentage of contacts with external nodes in the contact traces, finding similar conclusions in both cases would tend to indicate

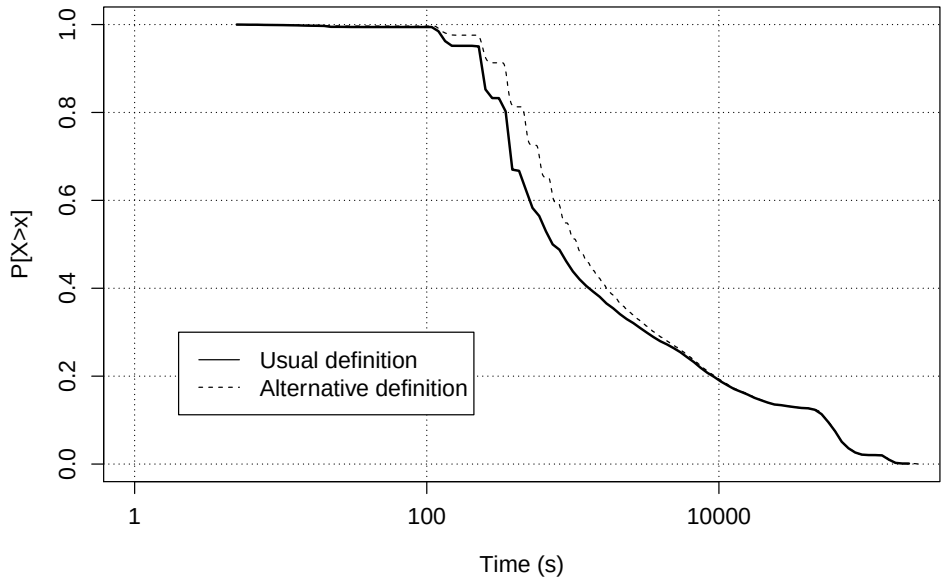


Figure 5.7: In uence of the use of the alternative ICT definition on the aggregated distribution for the Infocom 2005 dataset (log-linear scale).

that external and internal nodes exhibit close behaviors, regardless of the experimental conditions. More investigation on this is left as future work.

5.3.5 ICT alternative de nition

Last but not least, we now look at the ICT alternative definition introduced in Section 5.2.3 and on Figure 5.1. Recall that this alternative definition considers the intercontact to be the time difference between the *beginning* of two consecutive contacts. In other words, it equals to summing the contact time with the usual intercontact time. The results are presented in Figure 5.7 for the Infocom 2005 dataset.

From Figure 5.7, it seems quite clear that the impact on the ICT distributions is quite limited. As expected, intercontacts are slightly longer with the alternative definition. We find that the distribution with the smallest KS test distance D is in both cases the log-normal distribution. Furthermore, fitting results only indicate a small change in the estimated distribution parameters (= 7 33 and = 1 95 for the usual definition, versus = 7 57 and = 1 83 for the alternative definition). However, the impact on the contact times becomes huge. Indeed, if one wants to keep the total experiment time unchanged and at the same time use the alternative ICT definition, all contacts must become 0-second contacts. Note that this is actually a very common simplifying hypothesis in the literature on analytical performance modeling; it is also used in the derivation of the model introduced in Chapter 4. It has the additional drawback of making simulations impossible, because the ONE [47] does not perform any data exchange on 0-second

contacts. To solve this problem, two alternatives are possible. One can replace these 0-second contacts with an arbitrary value, as discussed in Section 5.3.1. It is also possible to keep the existing CT values, at the expense of modifying the total length of the trace.

5.4 Check-list proposal

Sections 5.2 and 5.3 have described the initial assumptions typically made before statistical analysis of a CT/ICT dataset, and their effect on the derived results in terms of statistical fittings. Although all hypotheses found in the literature make sense in the context of their paper, their diversity makes it difficult to compare results, and more often than not the reader is faced with the absence of hypotheses not in direct link with the paper goal.

In the remainder of this section, a checklist is proposed to keep the authors aware that apparently harmless assumptions might have a strong effect on statistical analyses and any subsequent result. For instance, Monte Carlo simulations could be impacted if these simulations rely on the fitted distributions. We identified three questions which are presented below.

Did I use the whole dataset? Most authors do some kind of filtering of the raw dataset they want to use, e.g., discarding inconsistent values (0-second contacts might for instance be considered as inconsistent) or values that do not fit the experiment they have in mind (e.g., very long contacts), or keeping only some periods of the whole dataset (weekends, etc.). This filtering should be carefully described: what data was filtered out? for what purpose?

Did I *really* use the whole dataset? Although one might believe the whole dataset has been used, the fitting method might implicitly filter out a lot of data. This is the case when setting the $x_{\min}$ threshold for the Pareto law. What happens to data samples lower than $x_{\min}$ should be stated: what data was left out? Should it be fitted in another way, for example using two different distributions such as exponential and Pareto? This question arises whenever a partial fitting is used. Similar observations apply to pairs left out of pairwise statistical analysis.

Did I change some values? Obviously, changing values in the dataset will have an impact. However, some statistical fitting tools, such as ML estimators, cannot be used with a dataset containing 0 values (70% of the Rollernet samples). Having such an amount of 0 values is a consequence of the shortcomings of the measurement devices used at the time of trace capture (e.g., Bluetooth scan time). Then the experimenter may have to explicitly choose between two possibilities: either keeping the dataset, and not fitting Pareto or log-normal; or accepting to slightly change the dataset despite the above warning (e.g., increasing 0-second contact times to an arbitrarily chosen value like 1 second). What happens to the surrounding ICTs in such cases is a choice left to the

researcher: leaving them untouched, thus making the experiment longer, or shortening them, which would mean changing even more values.

5.5 In uence on performance results

So far, we have restricted our study to the statistical analysis of aggregated distributions of CT. But this is rarely the final goal: in fact, we are ultimately trying by these means to correctly capture the general behavior of the network, so we can accurately predict afterwards the performance of applications running over it. Hence, this part aims to complete the previous study on the in uence of assumptions on distributions by extending it to the performance results.

5.5.1 Method

For this purpose, we needed to go back to the traces used to perform the statistical analysis. These filtered traces were then played in the ONE simulator [47], v. 1.4.1. Finally, the results were plotted using R [67]. Duration of simulation and number of nodes were constrained by the contents of the traces. For this section, we keep the same baseline assumptions as in Section 5.3. However and for comparison purposes, we will use the same message generation durations when considering varying trace lengths.

For all experiments, we applied epidemic routing. Size of the messages is set to 10 B. Although this value is very small, we repeated the experiments with a size of 100 B without noticeable changes on the results. Message TTL was set to the duration of the full trace to avoid message expiration, and buffer space on the nodes was set to 100 kB so it could be considered infinite with respect to the message size used, thus avoiding drops. A unique source periodically creates messages for a unique destination, and other nodes simply act as relays. Note that the source-destination pair is unique and fixed for the whole simulation. It is taken among the internal nodes of the corresponding trace. There is no background traffic. Due to the deterministic nature of all parameters (fixed period, source-destination pair, message size, buffer space; contacts constrained by the existing traces), we only need to run each simulation once.

The network performance metrics were chosen among the ones already available in report classes in the ONE: distribution of the delay (`MessageDelayReport`) and evolution of the delivery ratio over time (`DeliveredMessagesReport`). These metrics are also often used in the literature on DTNs.

5.5.2 Results

We now present the results for two traces: Rollernet [87] and Infocom 2005 [76]. For Rollernet, the source-destination pair is from node 27 to node 38, two nodes that were known to be respectively located at the front and at the end of the crowd. For Infocom 2005, which does not offer such a level of detail on participants, we arbitrarily choose to go from node 1 to node 41 because these are respectively the lowest and highest indexes among the internal nodes. We did not use the MIT dataset in this section, due to

its length which would have required more simulation time and a change in some of the hypotheses presented above (especially message TTL and message creation period). For Infocom, the message creation period is set to 1000 s, while we choose 100 s for Rollernet to get an acceptable number of messages because the trace is shorter.

From the five assumptions previously studied, we only keep three. We first remove the Pareto lower bound estimation technique because this assumption only impacts statistical results. Hence, it will not directly affect *performance* results, unless of course the statistical analysis is subsequently used to produce a synthetic contact trace. We also remove the ICT definition, because the ability to exchange data is mostly determined by contact duration. Furthermore, using a different definition would require modifications of the traces, thus going much farther than the other assumptions which are simply filters.

0-second contacts

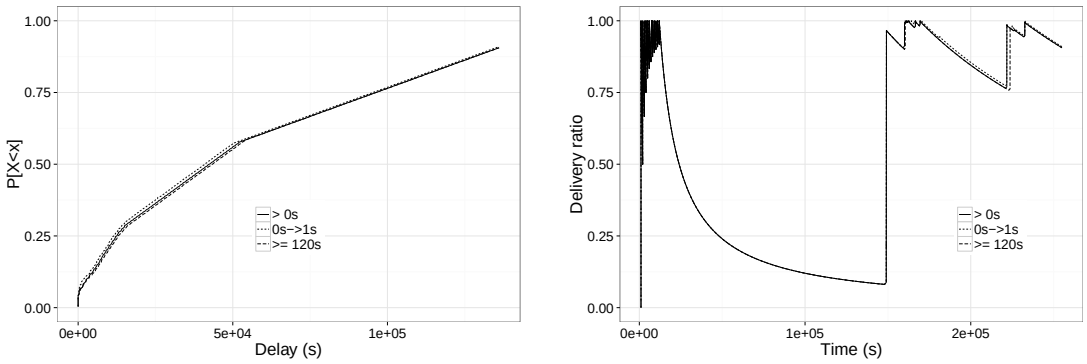
Like in the previous part of the study, we start with the 0-second contacts. We compare the same three possibilities: extension to 1 second (baseline), removal of 0-second contacts and removal of all contacts shorter than the sampling period. Note that with the simulator used, removal of 0-second contacts and playing the original trace with the zeros is strictly equivalent because no transfers can be scheduled during a 0-second contact. Again, the study is made on the 5000 first seconds of Rollernet and on the full Infocom dataset. The results are shown in Figure 5.8.

We do not observe any change on performance with the Infocom 2005 dataset, either for delay or delivery ratio. On the contrary, differences are clearly noticeable with the Rollernet dataset. Regarding the delay, extending 0-second contacts to 1 second offers a clear improvement compared to the removal, which is itself better than removing all contacts shorter than the sampling period. Considering the very high percentage of short contacts found in this dataset, which can for example be seen on Figure 5.2, this result was expected. The hierarchy of performance for the delay translates to the delivery ratio.

Trace length

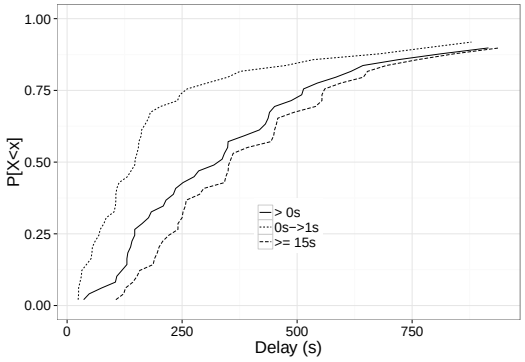
For this case, we only considered the Rollernet trace because this was the only one in our selection which had been used in part in previous works. Again, we compare the 5000 first seconds before the break and the full-length trace. In order to make a fair comparison, message generation only occurs during the 5000 first seconds in both cases. The results are presented in Figure 5.9.

In this case, we notice that the curves for the delivery ratio overlap for most of the experiment time. The time scale for the delivery graph only goes to 5000 s for both cases, because the trace offered sufficient connection opportunities to be able to successfully deliver all generated messages within the 5000 s time frame. For the delay, we observe a small reduction with the full length dataset. Here, the additional time brought by the

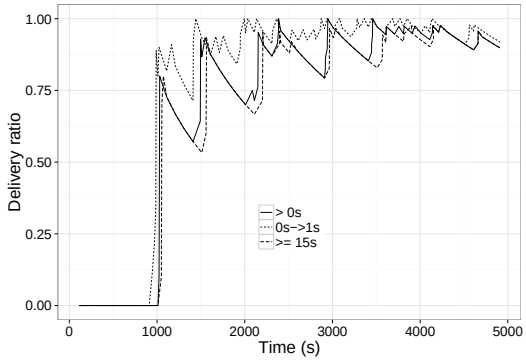


(a) Infocom - Delay

(b) Infocom - Delivery ratio



(c) Rollernet - Delay



(d) Rollernet - Delivery ratio

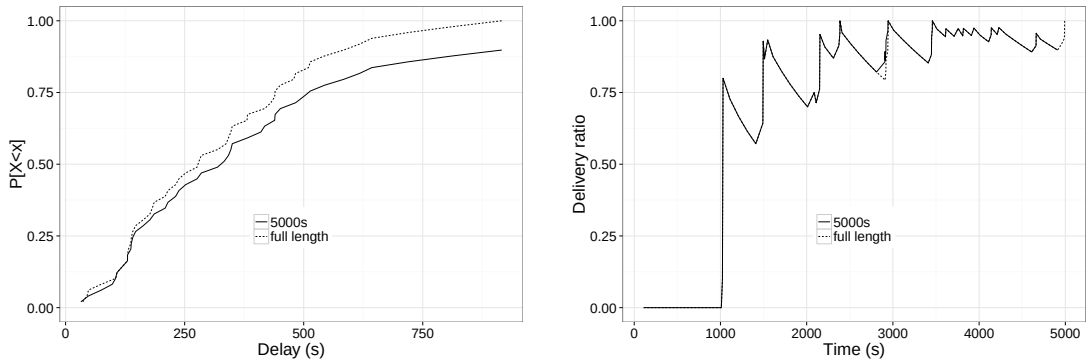
Figure 5.8: In uence of the 0-second contacts on network performance for the Infocom 2005 and Rollernet datasets. All scales are linear.

rest of the dataset also allows to reach a delivery ratio of 100%, which can be seen in both graphs.

External nodes

We now present the results for the in uence of the external nodes on the observed performance, in Figure 5.10. Because the source and the destination are both internal nodes, adding external nodes to the network only means adding extra relays and forwarding opportunities for the messages. Consequently, we can expect better performance when considering external nodes in the trace.

The results are this time contrasted between both traces. In the Infocom case, adding external nodes had absolutely no in uence on the performance metrics studied here. On the contrary, for the Rollernet case we can notice an improvement in both the delay and the delivery ratio. The explanation for this difference comes from the symmetric nature of the Rollernet trace: contact symmetry means that the external nodes can both send



(a) Delay

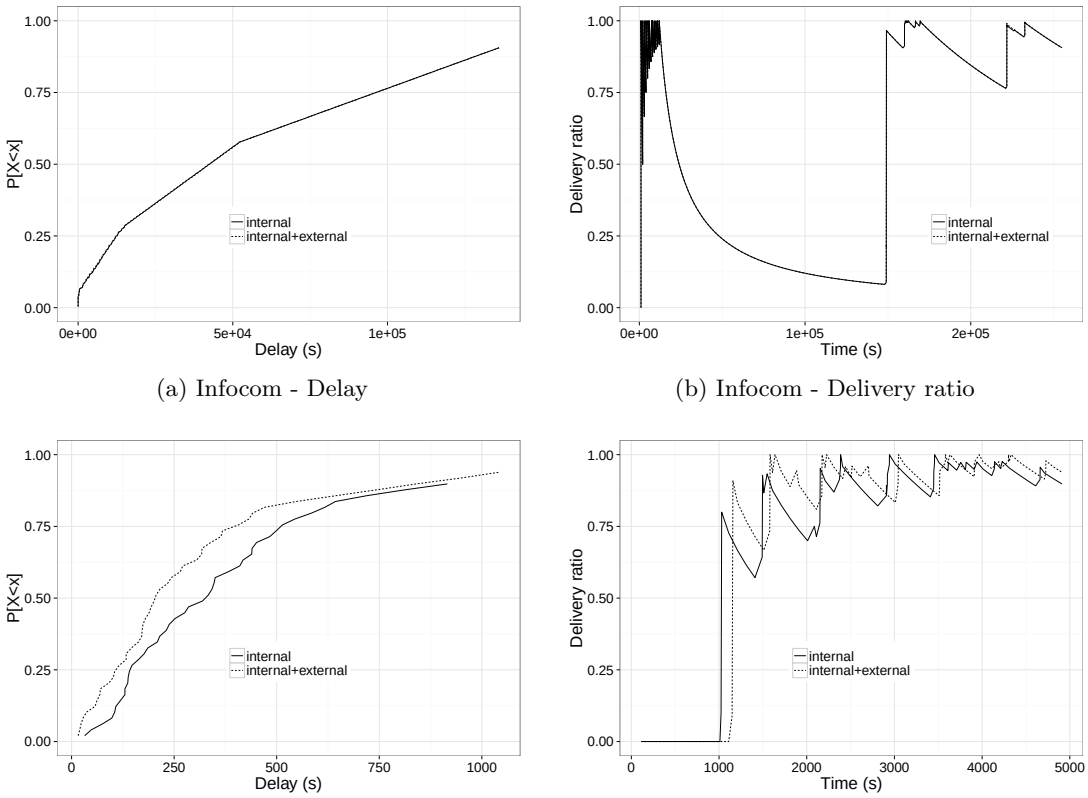
(b) Delivery ratio

Figure 5.9: In uence of trace length on network performance for the Rollernet dataset. All scales are linear.

and receive data, although they only received beacons during the trace collection and did not send any. When symmetry is not assumed, like in the Infocom case, the external nodes only receive data and never send it again, thus acting like dead ends for the messages. Such behavior could be extremely problematic for routing protocols without replication, as it means the only copy of the message could get trapped in one of these nodes and become irremediably lost. In the present case, we chose epidemic routing, for which the replication process allows to overcome this issue.

At first glance, the results found in the performance part seem to contradict the ones found in the statistical analysis part: in some cases, the differences found between statistical distributions are no longer visible when it comes to performance. One explanation for this is that the previous study was conducted on aggregated distributions, which are known to hide many details [61] and offer an averaged view of an otherwise rather contrasted situation. Another reason is that the simulator used, unlike distribution fitting tools, discards 0-second contacts. A second observation which can be made in this section is the in uence of the dataset chosen. The example of external nodes, for which conclusions are different between Infocom 2005 (no in uence at all, curves overlap) and Rollernet (moderate delay improvement), is enlightening, especially because the justification comes in fact from one of the preliminary filtering assumptions (symmetry).

As future work for this section, we could try to run the simulations with other parameters: different routing protocols, buffer sizes, node choice or message generation patterns. The choice of parameters remains an open question, due to the almost infinite possibilities here. We could also choose to consider other performance metrics, such as the number of hops or number of copies generated before message delivery. But a more interesting experiment would be to compare generated data from analyzed traces to the one from traces. Here, we only performed a performance analysis on the original traces with filtering. When conducting a statistical analysis, the aim is to generate a new,



(a) Infocom - Delay

(b) Infocom - Delivery ratio

(c) Rollernet - Delay

(d) Rollernet - Delivery ratio

Figure 5.10: In uence of the presence of external nodes on network performance for the Infocom 2005 and Rollernet datasets. All scales are linear.

synthetic trace with the same properties but a more suitable scale for the envisioned experiment. Comparing performance results of this new trace to the ones from original traces (instead of the aggregated CT and ICT values as in [83]) is necessary to ensure the interest of this statistical analysis approach.

5.6 Conclusion

In this chapter, we studied the statistical analysis process of contact traces. First, we summarized the preliminary assumptions which have been made prior to analyzing the data, based on previous works. Using three well-known datasets from the literature, we illustrated their in uence on the parameters of the fitted probability distributions of aggregated values. We showed that the treatment of 0-second contacts and the Pareto lower bound estimation strongly impact the curves, while trace length, ICT definition or external nodes play a smaller role.

We then turned to the study of the delay and delivery ratio with filtered traces, this

time considering only 0-second contacts, external nodes and trace length. We noticed that the conclusions derived from the statistical analysis part are not necessarily valid for the performance experiments, and clearly depend on the dataset used.

Considering that accurate models need to be derived from real data, and that we previously showed preliminary assumptions can strongly affect statistical derivations, this reduces the field of use for such models. More precisely, since a model would only capture the parameters of one precise situation, it would be unsuitable for generalization to other network setups or experimental environments. This represents another limitation of the use of contact datasets, in addition to the ones already mentioned in the literature.

As a matter of fact, these conclusions may be different for other datasets, as they are highly dependent on the exact nature of the datasets, and their usage in specific simulation/emulation setups. This is the rationale for building the checklist proposed here. This checklist is certainly not exhaustive, and should be expanded, at least with hypotheses described but not explored in this chapter. In the next chapter, we will try to address this limitation and go beyond statistical analysis to encompass the whole life cycle of contact traces.

Future work should cover the pairwise distributions to make a comparison with the aggregated distributions on which this chapter focused. It would also be interesting to conduct a performance study of contact traces generated from the statistical distributions studied here, and compare the results with those obtained using the original data. Finally, extending the performance study to other use cases, like for example a single-copy routing instead of epidemic routing, or other metrics, such as the number of hops, could also be valuable.

Chapter 6

From collection to scaling: on the use of traces for the study of DTNs

Contents

In the previous chapter, we focused on the hypotheses that could be taken when performing statistical analysis of real datasets. We also had a look at how these hypotheses in uenced performance results when directly applied to the traces.

However, this is only a small part of a bigger process. In this chapter, we are still interested in the hypotheses related to traces, but we now consider the whole process, from collection to statistical analysis and scaling. While the previous chapter mainly studied statistical distributions derived from the data, the focus here also encompasses other tools such as network performance, node degrees or raw data.

Note: this work already appeared in part in Elsevier Computer Communications under the title *Following the Right Path: Using Traces for the Study of DTNs*. http://dx.doi.org/10.1016/j.comcom.2016.05.006

Contact traces collected in real situations represent a popular material for the study of a DTN. Three main use cases can be defined for traces: social analysis, performance evaluation and statistical analysis. In this chapter, we perform a review on the technicalities of real trace collection and processing. First, we identify several factors which can in uence traces during collection, filtering or scaling, and illustrate their impact on the conclusions, based on our experience with four datasets from the literature. We subsequently propose a list of criteria to be verified each time a trace is to be used, along with recommendations on which filters to apply depending on the envisioned use case. The rationale is to provide guidelines for researchers needing to perform trace analysis in their studies.

6.1 Introduction

Recent years have seen a major growth in the number of mobile devices, almost all providing network connectivity through NFC, Bluetooth or WiFi to name a few. This has made both opportunistic and DTNs major topics of interest. Assessing the performance of such networks is a necessary step towards their deployment. So far, this task often relies on traces or datasets (we will use both terms indistinctly) including CTs and ICTs between nodes, coming either from synthetic models or captured in real-life situations.

At first glance, traces such as the ones available at CRAWDAD seem to be the most realistic material usable for DTN studies. There are however strong hypotheses captured into these traces: the capture setting (conference, campus. . .), the radio technology

(Bluetooth, WiFi), the number of nodes, the total duration, the scope (standard working hours vs round-the-clock recording), and the sampling period. Furthermore, because these data collection efforts are usually hard to perform, only few research teams have managed to provide such datasets. This led to some traces being used much more often than others. For the purpose of this study, we focus on datasets containing contact times between devices. GPS logs or network logs are consequently out of the scope of this paper.

In the previous chapter, we already studied the impact of some filtering techniques on the statistical analysis of contact datasets. We chose to focus on statistical distributions instead of network performance due to their use in some proposals in the literature [16, 83], aiming at scaling the captured networks in terms of time scale and/or number of nodes. We found out that among the filters used, some had a major impact on distributions, while the contribution of others was much more limited.

In this chapter, we try to go deeper in the study of traces. Hence, the work presented here considers all the steps in the life cycle of traces, from the real-life data collection to the statistical analysis. The various filters which can be applied in between are also discussed. The analysis of four existing datasets and their use in various research works provided us with several recommendations for future trace-based studies.

The chapter is organized as follows. In Section 6.2, we present the traces and identify use cases for which they can be exploited. Sections 6.3, 6.4 and 6.5 then provide a detailed inventory of all factors which could in uence subsequent results, when respectively collecting, filtering or scaling the trace. This in turn allows Section 6.6 to propose a list of parameters which need to be considered when working with a contact trace, and provide some advice. Then, in Section 6.7, we highlight and discuss some of the issues and open problems mentioned in the previous sections. We conclude in Section 6.8.

6.2 Background

In this section, we present the contact datasets which will be covered by our study. We also identify three main use cases for contact datasets, based on the existing literature.

6.2.1 Existing datasets chosen

We have selected four datasets to illustrate our study. Three of them (Rollernet [87], MIT [28] and Infocom 2005 [76]) were already used in the previous chapter and in several previous works, while the fourth one (Humanet [12]) represents a more recent experiment for which the authors made a careful study before choosing parameters. All are publicly available through the CRAWDAD archive website. We refer the reader to Chapter 2, where a more detailed description of these datasets is provided.

The four traces selected here offer a true diversity in terms of sampling periods, collection dates, trace lengths and captured environments. Although there are several other datasets available, the goal of the present work is to study the *assumptions* and not the datasets, which is why we decide to restrict ourselves to four of them.

Some studies of DTNs also used WLAN traces, such as the Dartmouth dataset [49]. In this paper, we decide to focus on Bluetooth traces; however, most of the issues highlighted here also apply to WLAN traces.

6.2.2 Typical use cases for traces

We already mentioned that real traces are a straightforward solution for DTN researchers willing to add some realism to their studies. More precisely, there are three major use cases for traces which can be identified from the literature: social and mobility inference, performance assessment and statistical analysis.

Social and mobility inference For this task, traces are typically processed in order to obtain social graphs, or at least links of various strengths (number of contacts, total duration of contacts, time of last contact, etc.) depending on the relationship between users. This in turn can allow to identify communities, or more simply to take routing decisions for protocols based on social proximity.

Performance assessment It is rather commonplace for a paper proposing a new routing protocol to demonstrate its applicability on both synthetic and real traces. In this case, real traces are used to overcome the potential lack of realism of the mobility models behind the synthetic traces. It was however shown in [73] that because contact traces do not record actual available bandwidth and buffer occupancy, they can lead to overly optimistic performance results.

Statistical analysis This was the main topic of Chapter 5, and is considered as a usual processing to go beyond the raw trace, for example by capturing the overall contact times distribution or node degree instead of individual values. This is also the approach used by [16, 83] in order to extend a trace in time span and/or in number of nodes, as detailed in Section 6.5.

6.3 Collecting the trace

In this section, we provide a list of parameters which have to be set at the time of recording. This list will be useful to both practitioners willing to collect new traces and researchers planning to exploit existing datasets.

6.3.1 Communication hardware choice

Traces such as Humanet, Rollernet or those of the Haggle experiments have been produced with custom-made devices such as iMotes. The choice of custom devices was made by the Humanet researchers to be able to tweak Bluetooth parameters which were not adjustable on smartphones [13]. For real applications, which are highly likely to be deployed on off-the-shelf devices, the additional constraints of such devices would need

to be taken into account. Constraints can come from the underlying operating system, the device usage patterns [81], or even, as mentioned before, the lack of control over some parameters. The MIT dataset [28] for example was produced using an app on smartphones, which were also used by participants on a daily basis to make phone calls or send text messages. The same year, researchers at the University of Toronto [81] also made a study on students equipped with PDAs running custom software. In these cases, exhausting the battery is not only harmful to the experiment itself, but also to the overall experience for the user. Note that this consideration would also be true for real applications. One parameter which can be adjusted in this case is the frequency to look for other devices, discussed next.

6.3.2 Sampling period

The traces considered in this paper have all been produced by probing at regular intervals for potential contacts. The choice of this frequency (called sampling period or sometimes granularity in the literature) can be determined by several factors. First, it is desirable to leave enough time for the other devices to respond to the probe. For this reason, the authors of [13] chose a value of 5 seconds after a careful study of response time for several smartphone models. Other limitations come from the portable nature of the devices used: a big sampling period has the advantage of saving energy (a concern already expressed in the previous subsection) and also memory [28]. Theoretical methods for choosing the optimal frequency can also be found in [59] and [66].

However, the choice of a big sampling period means that shorter contacts remain unseen, which may underestimate contact opportunities. To address this issue, the authors of [92] introduce a method to infer a *plausible* mobility from the contact traces, which can then be used to generate other traces with a higher temporal resolution. This approach was later criticized in [56], due to the high number of parameters which have to be set to properly capture the intrinsic mobility. We can also mention another solution of adaptive sampling algorithms, which vary the sampling period depending on the observed contacts or number of neighbors. Two examples are [40] and [90].

As an illustration, a close-up view of the distribution of contact times for the keynote contained in the Infocom 2005 dataset is provided in Figure 6.1. According to the conference program, this keynote lasted for 2.5 hours on the first morning of the conference. For clarity, we reduced the range of the x axis from 5000 to 1200 seconds, and of the y axis from 1600 to 250 contacts. Each bar has a width of 10 s. This plot only considers contact times for the keynote, but we also obtained similar shapes for other periods of this dataset (panels, sessions...). By using a histogram and linear scale instead of the much more common CCDF log-log plots, the sampling period becomes more visible: the time interval between each peak is 120 seconds, which corresponds to the sampling period advertised for this dataset. Notice that these peaks would translate into steps on a CCDF log-log plot. The fact that there are also other contacts whose durations are concentrated around these peaks comes from the desynchronization procedure, discussed in the next subsection. It should be noted here that the peak located at 0 second contains in fact 1575 contacts (instead of 250 as the scale choice might suggest), of which 1486

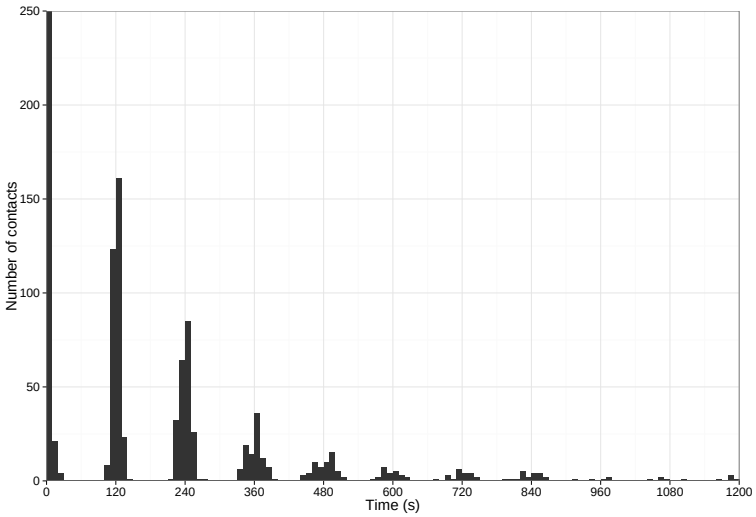


Figure 6.1: Histogram of the contact times for the keynote in the Infocom 2005 dataset. The sampling period of 120 seconds is clearly visible between peaks. The peak at 0 second contains in fact 1575 contacts.

effectively have a recorded duration of 0s.

6.3.3 Time synchronization

With a distributed data collection, it is crucial to have a common time reference to correctly detect encounters. Unfortunately, the intermittent connectivity which characterizes DTNs makes this challenging. Interestingly, perfect time synchronization may be undesirable during data collection, as two Bluetooth devices scanning at the same time cannot discover each other. For this reason, the authors of [76] and [5] used a random dephasing between device clocks, respectively of $12s$ and $5s$.

The authors of the Humanet dataset [12] used a nightly synchronization: each night, when the collection device was left at the office for charging and data uploading, it received a new timestamp from the central server [13]. During the day, perfect synchronization of scanning was avoided by using a random slave period $(3 + rand(1\ 5)$ s). On the contrary, authors of [76] and [5] performed a manual synchronization after the experiment, a process which is prone to errors: for example in [76], accuracy below 5 mn is not guaranteed.

6.3.4 Storage format

At the end of the experiment, data from all participating devices is gathered and merged. Then, the resulting files can be used for evaluation and/or shared with the community. As was already noticed in [58], there is no common format for all the traces. Some are available as text files [76, 5], others as SQL database dumps [28, 12].

The type of data stored also differs: if it is always feasible to find the identity of the two nodes involved, along with the start and end times of the contacts, contact and intercontact times sometimes have to be computed. Format of times also vary, ranging from Unix timestamps [28] to seconds from the beginning of the experiment [76]. Consequently, a researcher willing to use several traces would first have to decide for a common format, then do some scripting to get all the traces in this common format.

For access point records such as [49] (which were not used in this survey), the trace does not directly contain contacts between nodes, but only connections to access points; hence, a common assumption is to consider two nodes as in contact when they are both simultaneously connected to the same access point (see for example [23] or [85]).

6.4 Filtering the trace

In this section, we now place ourselves from the point of view of a researcher who would like to use real traces for his work, such as datasets available at CRAWDAD. Although it seems straightforward to use the raw data directly, there are in fact several filters which can be applied. Notice that this section can also be valuable for users of synthetic traces, because it lists several parameters of interest for simulations.

6.4.1 De ning intercontact times

A first factor which needs to be accounted for is the definition of an ICT, as we already pointed out in Chapter 5: while most of the literature considers it to be the duration between the end of a contact and the beginning of the following one, some papers such as [83] consider the time difference between the beginning of two consecutive contacts. Because this latter definition aggregates both the contact time and the usual intercontact time, we believe it to be a reasonable choice when contacts are considered instantaneous, as is often the case in theoretical performance models like those presented in Section 2.2. Due to these two definitions, special care must be taken when ICTs have to be computed from contact start and end times, like we mentioned in the previous subsection; to the best of our knowledge, precomputed intercontact times available in some datasets already follow the usual definition.

6.4.2 Symmetrizing contacts

Bluetooth uses an asymmetric discovery procedure, which means that a recorded contact between node i and node j does not necessarily mean that j and i could also communicate; in fact, the authors of [94] claim that this is rather uncommon. However, some works choose to assume symmetry [87]. For social analysis, symmetry is likely to be desirable (if user A meets user B, then B also meets A). However, issues can appear with symmetric contacts when computing node degrees. In this case, researchers should not forget that a contact between nodes i and j means that j contributes to the degree of i, just as much as i contributes to the degree of j. We show the differences for the case of the internal nodes of the Rollernet dataset in Figure 6.2. Here, forgetting that the

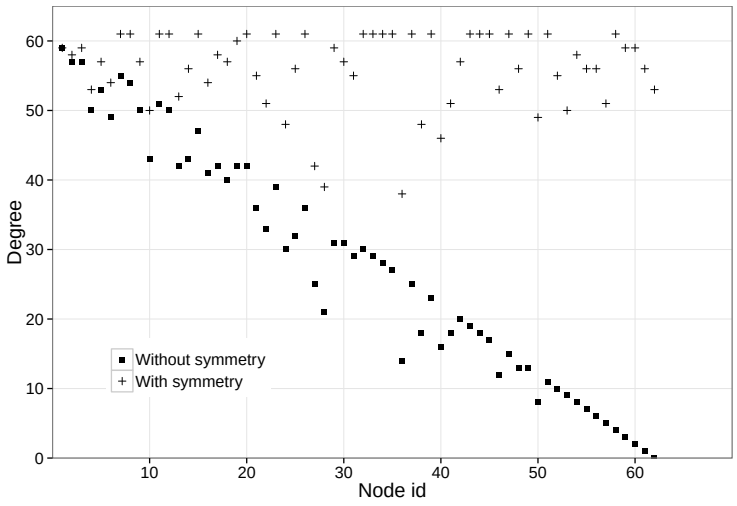


Figure 6.2: In uence of the symmetry on the degree computation for the 62 internal nodes of the Rollernet dataset. In this example, forgetting the symmetry underestimates the degree, especially for the nodes having the biggest indexes.

raw dataset assumes symmetry leads to an underestimation of the computed degrees, especially for the nodes with the biggest indexes. Consider for example node 58 (out of 62), which degree jumps from 4 to 61 when symmetry is correctly taken into account. This leads the authors of [89] to conclude that node degrees exhibit a linear distribution, while the results presented here in Figure 6.2 with symmetry clearly contradict these findings.

On the contrary, symmetry should preferably be avoided for network performance assessment, unless of course the connection is effectively symmetric.

6.4.3 Merging contacts

In contact traces, a contact between nodes i and j lasts as long as probes from i get an answer from j. However, the authors of several datasets observed a large number of contacts which were only separated by one probe, and consequently decided in this case to merge such contacts. For the Infocom 2005 dataset, this choice is made to address a memory exhaustion problem caused by contacts with a specific brand of mobile phones [43].

However, this strategy can merge two distant contacts, thus creating an artificially long contact which may not correspond to the reality. Although this may be desirable when inferring mobility or social relationships from the trace, it would lead to overestimating the transfer abilities of the network, thus biasing performance evaluations.

To illustrate this behavior, we applied such a filter to a synthetic trace. The trace was generated with 41 nodes on a 100 100 m^2 area, moving according to the Random Waypoint model (constant speed of $1m\ s$, no pauses) during 2.5 hours, the first one

being discarded to allow convergence of the model. Node range was 10 m. The original sampling period was 1 second, a small value achieved thanks to the use of a simulator (namely, the ONE [47] v1.5.1) which would not be possible in reality. The final sampling period applied by the filter was 120 seconds. Note that the choice of parameters (node number, duration, sampling period) was directly inspired from paper sessions in the Infocom 2005 dataset.

The results are presented in Figure 6.3. In the original trace (Figure 6.3a), the longest contact recorded had a duration of 219 s. On the final trace however (Figure 6.3b), we notice the appearance of 28 contacts with a duration of 240 s, which did not exist in the first trace, and even a 360-second contact. When evaluating performance, such long contacts would be seen as highly interesting data transfer opportunities, despite being only artifacts of the contact merging process.

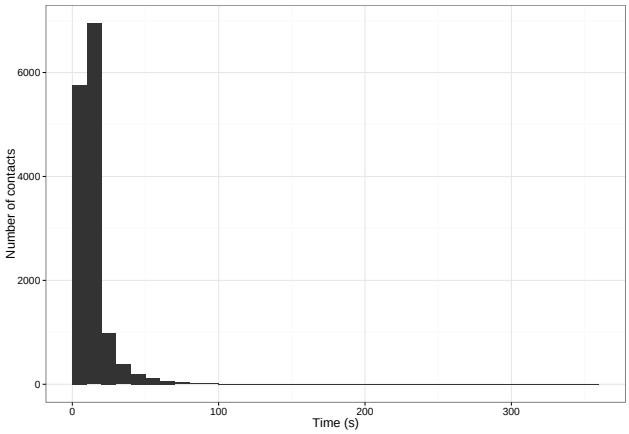
One may however object that the previous study was done with a synthetic dataset instead of a real one. In fact, a similar study was also conducted in [14] for the Humanet dataset. Originally produced with a sampling period of 5 seconds, the authors processed the trace to virtually achieve sampling periods of 120 seconds as in [76] or 300 seconds as in [28]. Similarly to the present work, increasing the sampling period produced longer observed contacts, and dramatically reduced the total number of contacts recorded.

Like the previous filter, this one is sometimes already included in the traces available on repositories: this is the case for example for Rollernet (symmetry, contact merging) [5] and Infocom 2005 (contact merging) [76]. This can be problematic because for such datasets, there is no straightforward way of assessing the impact of these choices due to the unavailability of the raw, unfiltered data; worse, users of such traces may not even realize the implications of this filtering. On the contrary, the more recent Humanet dataset [12] does not include any filter in the released version; some are however explicitly applied by the authors for their subsequent studies [14].

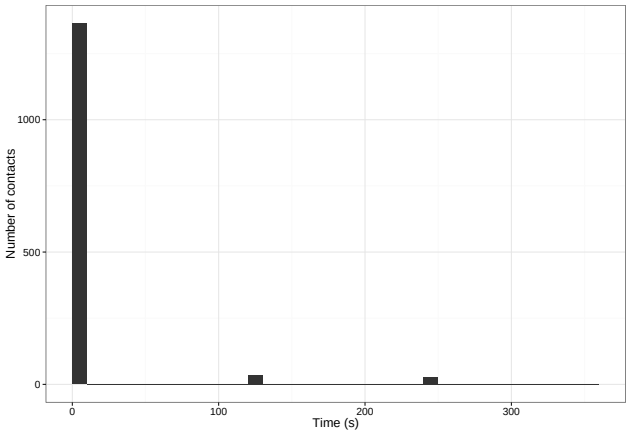
6.4.4 Losing or removing extreme values

The range of values recorded in contact datasets can be quite broad, with intercontacts ranging for instance from a few seconds to a few days in the Infocom 2005 dataset. Hence, one may want to remove values which are outside a certain range. Some of them can indeed be seen as a kind of artifact of the collection process: in the case of the Infocom 2005 dataset, the longest ICTs last for almost three days, which happens to be the length of the whole experiment. They could therefore be interpreted as pairs of nodes which saw each other only at the beginning and at the end of the experiment, thus indicating that the corresponding people would have been unlikely to meet at all without the experiment. It should also be noted that the disappearance of the longest values can be a side effect of restricting the length of the trace: it is indeed impossible to record durations longer than the trace length, as was already noticed in [29].

However, extreme values can also be the smallest ones. Among these, one value is especially interesting: contacts with a duration of 0 second. They represent a large part of the raw Bluetooth traces studied here (75% of Rollernet, 52% of Infocom 2005, 43% of the MIT dataset and 55% of the Humanet dataset). It should be noted that these



(a) Original trace



(b) Filtered trace

Figure 6.3: Distribution of the contact durations in the original and synthetic traces. Note the appearance of longer contacts after the filtering.

contacts did not really last for 0 second during the trace collection; instead, they were simply shorter than the sampling period and were only detected for one probe. This can be seen for example on Figure 6.3 in Section 6.4.3, where the application of the sampling algorithm led to a majority of recorded contacts with a duration of 0 second while the original trace did not contain any (minimum duration was 1 second). Recall also the highest peak at 0 second in Figure 6.1. Yet, some papers such as [83] appear to discard all contacts shorter than the sampling period (including the 0-second contacts), thus removing a substantial part of the data. On the contrary, the authors of [87] choose to include these 0-second contacts in their analysis by extending them to an arbitrary value of 1 second. This extension, despite being far from reality, is however necessary for some statistical tools due to the presence of logarithms in the formulas, as mentioned in Chapter 5 which uses ML estimators.

The question that is raised by these choices is about their impact on the intercontact times: if some contacts are deleted, then the surrounding intercontacts should be merged because they no longer separate anything. This would lead to overestimating the intercontact length.

6.4.5 Restricting contacts to device mobility

Referred to as the human mask in [14], this filter is used to discard contacts which happened while the device was not worn by the user (e.g., it was idle on the desk). If the use of this filter is desirable for inferring mobility from the contact traces, it should be avoided when evaluating network performance, since it may discard useful connections. Furthermore, applying this filter requires a way to tell when the device is worn and when it is not: in the case of the Humanet dataset [12], this was achieved through the collection of accelerometer data, a parameter which is unfortunately not present in all traces. For the MIT dataset [28], for which no accelerometer data was available, the authors tried to implement a so-called 'forgotten phone' classifier to identify times when the phone was not with the user.

6.4.6 Filtering devices

Traces may not only record contacts between devices participating in the experiment (internal nodes), but also with other devices which were observed despite not collecting any data (external nodes).

The issue with the external nodes is that they can be detected, but not detect others. In other words, and unless symmetry of contacts is assumed, they will be able to receive data, but not to send it. Consequently, and for simplicity, contacts with these external nodes are sometimes simply discarded from the trace. The study of aggregated distributions of CTs and ICTs led the authors of [43] and later of this thesis to the conclusion that both categories of nodes exhibited similar behaviors.

Even when the nodes are restricted to the internal ones, further filtering may be needed. Some datasets ([12] and most of the traces in [76]) involve fixed nodes, typically placed by the researchers in strategic zones where people are highly likely to

meet. Unlike the others, these nodes are not tied to a person. While these nodes are especially interesting for location inference, they are also known to dramatically improve the performance of DTNs [4]; hence, they should be either properly acknowledged or discarded when evaluating network performance on such traces. For statistical inference or mobility modeling, we believe that both external and fixed nodes should also be discarded, because they will typically exhibit different connection patterns.

6.4.7 Pair discarding

To ensure statistical validity of the distribution fitting process, which will be discussed in more details in Section 6.5, we need to have a minimum number of samples for each pair. This lower bound can also be interesting when building social graphs, as a way to remove the less active pairs from the analysis which would translate into low-weighted edges and unnecessarily clutter the graph. The authors of [23] use a threshold of 4 contacts, while the value chosen in [87] is 9 contacts. It should be noted here that the assumption of symmetry mentioned earlier will also impact the number of pairs which would be discarded. Indeed, merging the contacts from node i to node j with those from j to i into a single pair will logically increase the total number of contacts recorded for this pair, eventually going over the threshold. Hence, the use of the symmetry filter should be properly acknowledged for such uses, as was done in [23, 87].

The major issue of this filter is the fact that some pairs will be removed from the analysis, as if they did not exist. This is especially problematic for statistical analysis, because it means that such pairs being missing as inputs will typically be also missing in the data generated from this analysis. One possible solution could be to aggregate the data from all such pairs, and treat them as one big virtual pair, at the expense of ironing out any differences that may exist between the original pairs. This will be discussed again in Section 6.7.

6.5 Scaling the trace

Once the trace has been adequately filtered, this may not be sufficient for the envisioned purposes. Indeed, datasets may contain time periods which exhibit different properties: some nodes may not be functioning properly [28], or the experimental conditions were different (such as a break during a rollerskating tour [5], holidays between school terms [28], or nights [76]). Sometimes, the dataset is simply too long for the envisioned experiment, so that only a subset of it suffices. This subset should however be chosen carefully: it would be unfortunate to try to study the dynamics of users in a session of the Infocom 2005 conference by taking a 1.5-hour[1] long sample in the middle of the night, or studying the on-campus interactions of MIT students over a holiday week.

But the opposite can also happen: the trace can be too short in time or not contain enough nodes. Due to the inherent difficulties of setting up trace collections, and the resulting limited number of existing datasets, such limitations are commonplace. To

[1] This was the session length, as found in the conference program.

address these shortcomings, two solutions have been proposed in the literature, namely the Community Trace Generator [16] and the Encounter-based MOdel [83].

6.5.1 Existing scaling approaches

These two proposals share a common approach: extracting characteristics from the trace as statistical distributions (and not as raw data), which are subsequently used for the generation of a new trace exhibiting the same statistical properties. CTG [16] captures the number of nodes along with the distributions of node degree, aggregated CTs and ICTs. EMO [83] also captures the number of nodes and distribution of node degree, but contact and intercontact times are treated in a pairwise manner. More precisely, the pairwise empirical distributions are first fitted to a probability distribution (which needs to be the same across all pairs), then the empirical distribution of the *parameters* of these distributions is fitted to another probability distribution. For example, the pairwise contacts of the MIT dataset [28] are found to be exponentially distributed, with parameters of the exponential law following a normal distribution [83].

At this point, we would like to mention again the aggregated contact time distribution shown in Figure 6.1. Because of the sampling process, contact durations are distributed close to the sampling periods, leading to ties (several samples with the same values) and an intermittent shape, with long runs of time values for which no contact is recorded. This is in sharp contrast with the continuous character typically displayed by usual statistical distributions.

6.5.2 Statistical tting issues

A first problem comes from the distribution fitting process used, which strives to use the *same* statistical distribution to fit the (inter-)contacts of all the pairs. In [23] and later in [87], it was shown for four different traces (Infocom 2005, Rollernet, MIT and Dartmouth) that the ICTs of some pairs could be modeled equally well by several distributions, while for some other pairs none of the three distributions considered (namely Pareto, exponential and log-normal) gave a satisfactory result. Interestingly, the distribution which was able to represent the largest number of pairs was for the four traces the log-normal distribution, despite the variety of environments captured.

We applied this statistical fitting approach (one distribution to model all the pairs) to the pairwise ICTs of the Infocom 2005 dataset [76]. We provide the list of parameters for the filters introduced in the previous section in Table 6.1.

No extreme values are discarded. As shown in Table 6.1, 0-second contacts are however extended to 1 second, so no intercontact has to be modified. We used the same statistical fitting procedure as in the previous chapter (Maximum Likelihood estimation for the parameters, and KS test to determine the best candidate). The choice of distributions to test is also the same: exponential, log-normal or Pareto with $x_{\min} = 1$ (value choice is arbitrary).

From the three distributions considered, we found out that the log-normal one gives the lowest distance D for 89.7% of the pairs. 6.4% have Pareto as closest distribution,

Table 6.1: Filter settings for the fitting study.

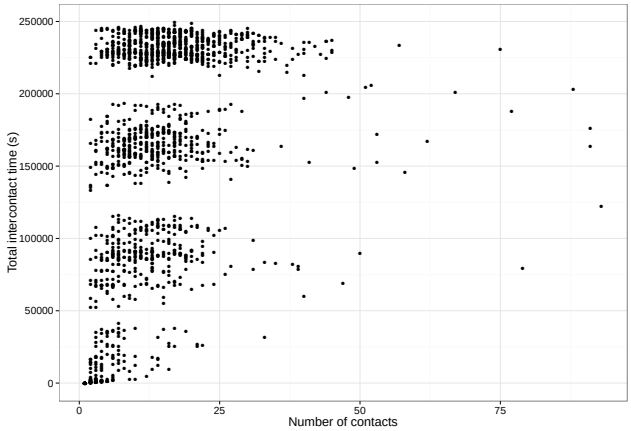
Filter	Value
ICT definition	usual
Symmetry	no
Contact merging	yes, included in raw trace
Extreme values	0 second contacts extended to 1
Device mobility	NA (no data available)
Device filter	internal nodes only
Trace length	whole trace
Contacts per pair	3

and 3.9% are closest to the exponential distribution. In [23, 87], the authors also found log-normal as the best candidate, using the Cramer-von Mises test and 9 contacts as a lower bound for the symmetrized pairs. When considering the p-values also provided by the KS test, log-normal remains the best candidate but with only 51.7% of the pairs for which the hypothesis cannot be rejected ($p > 0\ 05$). However, we found that 45.6% of the pairs considered were *not modeled by any of the three distributions considered*. Possible explanations for this include the small value chosen for the lower bound in the number of contacts for a given pair. These unfitted pairs also have in common a few intercontact times whose values differ from at least one order of magnitude from the other values recorded for the pair.

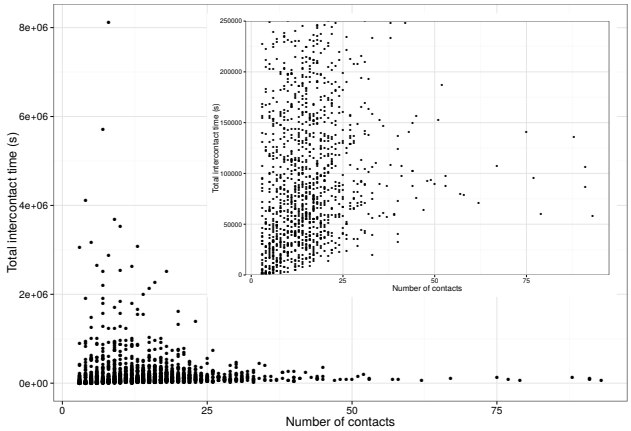
Then, we used the estimated parameters found for each pair with the log-normal distribution to generate the same number of synthetic ICTs for each pair as in the original trace. The results in terms of total ICTs can be found on Figure 6.4. On these plots, each dot represents a node pair. When comparing the original trace with the generated one, we notice that the totals for some of the pairs are much higher as the original ones (even exceeding the initial trace length), implying that longer intercontacts have been generated.

Furthermore, the dots for the original trace are grouped into four regions. Closer inspection of the intercontact times for each pair reveals that this structure is caused by a few very long intercontacts (having durations of at least one night). These long (but rare) intercontacts, which could be seen as extreme values, are lost after the fitting process.

Even the fitting process itself can hide many details. A first example is the Pareto law. In [61], the authors mention two definitions for it, which they name Pareto0 and Pareto depending on their ability to accept values arbitrarily close to 0. When only the Pareto law is used, it is also possible to find several definitions for the exponent. Another issue with power laws is the fact that they have a lower bound, usually written $x_{\min}$. In Chapter 5, we already mentioned that there are two possibilities to estimate this lower bound: arbitrarily setting it to a value such as the sampling period [83], or using a mathematical estimator such as the one proposed in [22]. In both cases, data below the lower bound will be discarded, as if the filter presented in Section 6.4.4 had



(a) Original trace



(b) Synthesized trace

Figure 6.4: Sums of pairwise intercontact times, for both the original dataset and the synthetic one obtained through statistical analysis. Each dot represents a pair. Some pairs exhibit much longer durations in the synthetic trace than in the original one, and the four-group structure disappears. Notice the scale change between the two graphs.

been applied. This makes comparison between distributions somewhat trickier, because other statistical distributions (like, for example, the exponential one) do not have any lower bound in their usual definition, and will therefore try to fit all the data. Once the various distributions to test have been chosen, a tool to help decide which one best models experimental data is needed. As we mentioned in Chapter 5, there are several possibilities. The simplest is to plot the data with an adequate scale and perform a graphical fitting, as in [43]. Other authors used statistical tools, such as the KS test [83] or the Cramer-von Mises test [23, 87]. Indeed, according to the authors of [22], statistical tests should always be favored over graphical fitting because the latter approach is too error-prone.

6.5.3 Validation issues

These statistical fitting approaches have another drawback: they are validated by showing their ability to reproduce aggregated distributions of contact and intercontact times. The major advantage of aggregated distributions over their pairwise counterparts is that they are much more compact (only one distribution to consider) [23], which made them a common approach in the literature for validation. However, the authors of [48] claim that validation should be made on parameters which are *not* used as inputs of the algorithm: they validate their mobility model (which is based on hotspots, speed distribution of users and transitions between hotspots) by its ability to correctly estimate the number of people at a hotspot across the workday.

Furthermore, aggregated distributions are known to hide many details. In [23], it was found that both exponential and log-normal pairwise distributions could lead when aggregated to the observed power laws. The authors of [61] also show that several other pairwise distributions can lead to power laws when aggregated, and provide a list of cases when using only the aggregated distribution is in fact not correct. Another issue was shown in [85]. In this work, the authors select three mobility models, and tune their parameters to reproduce the empirical aggregated distributions of CTs and ICTs found in real datasets. When performing simulations with both the real traces and the synthetic ones derived from them, they find out that synthetic traces always provide much more optimistic performance results than the real ones.

6.6 Recommendations

In Section 6.3, we reviewed the parameters to be considered at the time of data collection, while Section 6.4 provided an overview of the filters applied for trace exploitation. Then, Section 6.5 presented two approaches to scale existing traces and their limitations. Based on these findings, we can now list the various parameters which have to be considered when using a trace.

6.6.1 Production

First, the hardware and radio technology used for collection must match the ones which are envisioned for the application. The sampling period must also be chosen carefully, taking into account both what is planned for the application and the limitations of the radio technology, as detailed in [13]. More precisely, the sampling period must consider the time to find other devices, but also the acceptable impact in terms of energy consumption. However, energy is no longer a major concern: while the team behind the MIT dataset [28] tried in 2004 to make cell phones batteries to last for more than 2 days without charging, it is now not uncommon to charge a smartphone on a daily basis. Based on the conclusions from [14] and our complementary analysis on contact merging, a realistic trace should also be as precise as possible, thus requiring a small sampling period. Therefore, a trade-off between energy and precision is needed. We believe that trace collection efforts should focus on precision, at the expense of frequent charges of

the experimental devices; on the contrary, real application deployments should aim for a pleasant user experience, which means limiting their footprint on energy consumption.

Issues on time synchronization must definitely be considered; based on existing literature, we would recommend random slave periods [13] instead of clock desynchronization, due to the smaller resulting errors.

If the collected data is to be made public, it is desirable to perform as little filtering as possible on the data, which will then allow other researchers to apply their own filters depending on their goal. Regarding these filters, their application should always be properly and precisely described, especially when they are incorporated in released data (as has already been the case in some datasets for symmetry, intercontact definition or contact merging). The aim here is to make results easier to reproduce and compare, a concern which was the original motivation for Chapter 5.

6.6.2 Filtering

A summary of recommendations on the filters can be found in Table 6.2, for the three common uses of traces introduced in Section 6.2.2: social structures and mobility inference, performance evaluation, and statistical analysis. The filters are presented in the same order as they were introduced in the previous sections. For each filter, we detail for each use case if it CAN, MUST or MUST NOT be applied. In this context, *can* means that we believe it is possible to apply this filter, provided that the researcher is well aware of it and of its consequences.

For example, the length filter can be used in all cases to isolate a part of the trace exhibiting particular properties (workdays, coffee breaks...). The same observation applies to the ICT definition: using the alternative definition is not wrong by itself, except when simultaneously considering non-zero contact times. Regarding symmetry, we also believe that it can be applied if the connection was really symmetric, but must be avoided otherwise.

For the other filters, the situation is slightly different. Merging contacts is recommended for social and mobility inference if one does not want to minimize the links between nodes in the network, as also found in [14]. On the contrary, overestimating data transfer opportunities will be detrimental to performance assessment and statistical analysis. We recommend filtering (or modifying) extreme values only for statistical analysis, at the very least when a large percentage of 0-second contacts is present in the trace. Similarly, we recommend the use of the mobility filter only for social or mobility inference, because it would underestimate transfer opportunities in the two other use cases considered. Regarding the choice of devices to consider, a study on social or mobility inference should typically ignore fixed or external nodes because they do not represent social interactions. Fixed nodes may however be useful for localization. For performance assessment, the choice of nodes is left to the researcher, as long as it is properly documented. For statistical analysis, the choice of nodes should also be carefully made in order to avoid the capture of unwanted behaviors. One may object that conclusions of [43] and Chapter 5 indicated similar trends among both internal and external nodes; however, these conclusions did not consider fixed nodes, and were drawn based

Table 6.2: Recommendations for the use of filters. + = MUST, - = MUST NOT, o = CAN.

	Social/mobility inference	Performance assessment	Statistical analysis
ICT definition	o	o	o
Symmetry	o	o	o
Contact merging	+	-	-
Extreme values	-	-	+
Device mobility	+	o	o
Device filter	+	o	+
Contacts per pair	-	-	+
Trace length	o	o	o

on the study of aggregated distributions instead of pairwise ones. Finally, ensuring the validity of the statistical analysis requires a lower bound on the number of contacts for each pair, a condition which we believe is less necessary otherwise.

It should also be noted that there can be some interplay between the filters: for example, symmetry will increase the number of contacts for a given pair, and, due to its impact on ICTs, also modify the range of values; choosing only a portion of the dataset will cause all (inter-)contacts longer than this new duration to disappear; removing the shortest or longest values directly in uences the number of values of the node pair in question, as does the mobility filter or the trace length. This list is not exhaustive, and represents an even stronger motivation to properly document the filters used in a study.

However, one may argue that if a researcher has enough control on the experiment to correctly set all the parameters mentioned above, it may be desirable to skip the trace collection part and directly proceed to application deployment. But even in this situation, trace collection can be a reasonable choice, since a recording of all contacts will be available for subsequent trials. Availability of this data would increase the repeatability of the deployment, or allow prior experimentations on an emulator[2]. It also allows to exploit the trace with various filter combinations, and then possibly for multiple use cases. For example, it would be possible to increase the sampling period without compromising the validity of the collected data.

This raises the question of representativity of traces. More precisely, real applications are unlikely to be limited to simply probing their surrounding for other nodes, which is what trace collection efforts have been doing so far; instead, they will also involve data transfer between devices (of images, news items, chunks of large files...). Assessing the difference in terms of recorded contacts between both approaches is outside the scope of this work, but would represent a highly valuable result for the community. A first step in this direction has been made in [73], where the authors recorded message exchanges in addition to the contacts between nodes.

[2]The fact that the next part of the thesis describes such a system is obviously purely coincidental.

6.6.3 Scaling

We mentioned that available traces do not always exactly match the situations one is willing to study. In Section 6.5, we presented some solution proposals to this problem. We do not see any issue in taking only a subset of an existing dataset, provided the choice of time period is carefully conducted (i.e., not mistaking a conference session with a lunch or a social event) and precisely described. If already present in the trace, ICTs should be recomputed to re ect the time span change. However, based on the various limitations previously outlined, we would strongly advise against upscaling (extending the number of nodes and/or the time span), unless new tools are created to address current issues of the existing approaches. Based on the findings expressed in Section 6.5, these new tools should be able to work at the pair level (and not only with aggregates [16]), and be able to use several statistical distributions to model them instead of only one. Definitions of the statistical tools (distributions and goodness-of-fit tests) should be properly provided to avoid any ambiguities when building upon the results. Furthermore, the quality of such tools should not be judged on their ability to accurately reproduce aggregated distributions [85], but rather on pairwise distributions or network performance results. In all cases, they must not be solely validated on their ability to reproduce network characteristics which are passed as inputs, such as in [83].

6.7 Discussion

The previous sections dealt with surveying the various filters, and providing recommendations on their use. In this section, we try to open new perspectives and summarize some of the questionings that were previously mentioned in various parts of the chapter. We can split them in two parts: the hypotheses taken either at the collection or at the filtering stage, and the choices to make during statistical analysis.

6.7.1 Hypotheses

Which de nition of ICT should I choose? This was partly discussed in Section 6.4. Although theoretical performance models do consider instantaneous contacts for simplicity, real contacts typically exhibit a finite, non-zero duration which impacts on data transfer abilities. Consequently, the ICT definition should be chosen depending on the desired contact time duration: simultaneously considering the alternative ICT definition and non-zero contact times equals to lengthening the trace, which may not be the desired outcome.

Which sampling period to choose? We already pointed out the necessary trade-off between a small sampling period and the associated increased power consumption that a real application would have to consider. For trace collection experiments, we believe that precision should be favored over energy consumption in order to make the resulting data as widely applicable as possible. More precise recommendations are however out of the scope of this chapter. A good example of a choice process can nevertheless be

found in [13]; however, the approach taken by the authors of evaluating response time on several smartphone models is unlikely to be possible in reality, due to the tremendous number of smartphone models available on the market. Also note that here we did not even mention the variety of software and operating systems.

How to deal with extreme values? Extreme values were already mentioned in the checklist presented in Chapter 5. We also saw in Section 6.5 that the longest intercontacts, despite their rarity, were sufficient to make the statistical method used fail at correctly capturing the whole structure of pairwise ICTs. 0-second contacts are even trickier. Not only are they the smallest value possible, they also have the unpleasant drawback of playing quite badly with formulas using logarithms like ML estimators. They are even paradoxical at first glance: 0 second means that nothing can be transferred (as would happen when playing the trace in a simulator), but these contacts do appear in the final trace, meaning that some data had been transferred during the experiment. The real interpretation for this comes in fact from the periodic sampling process, as explained in Section 6.4.4. In Chapter 5, we mentioned three possibilities to handle these contacts: removing 0-second contacts, extending them to an arbitrary value or removing all contacts shorter than the sampling period. We cannot recommend a specific approach over the others; however, it should be noted that data removal has the largest impact among the three, due to the required modifications to the intercontacts. In the next subsection, we consider statistical analysis. Interestingly, none of the distributions curretly used for DTN studies can handle the observed peak at 0 second. However, it should be noted that a possible workaround could be to use an approach similar to *zero-in ated models*. We would give a probability for the value to be equal to zero, and provide a distribution for the remaining cases when it is not zero. More investigation on the mathematical soundness of this approach is left as future work.

6.7.2 Statistical analysis

We now turn to the challenges and questionings related to the statistical analysis process.

How to deal with deleted pairs in statistical analyses? The statistical analysis is only valid when a sufficient number of samples is available. For real traces, we mentioned that this requirement translates into deleting some pairs when their number of recorded contacts is below a given threshold. What happens to these pairs after the analysis process is still an open question: should they be used as-is in scaled traces? Should they be all merged together to produce a virtual pair with a sufficient number of contacts to ensure statistical validity? Should they be simply left apart?

How to deal with *unfitted* pairs in statistical analyses? Deleted pairs are not the only issue with statistical analyses. Most of the time when conducting trace analyses, the experimenter will have to face the issue of some node pairs which, despite having a sufficient number of contacts, cannot be satisfactorily modeled by any of the distributions

chosen. In Section 6.5, recall that 45.6% of the pairs were unfitted. This high value is a good reason to consider this issue seriously. At this point, several approaches are possible:

additional distributions can be introduced, in the hope that one of them will prove to be a better candidate. In [83] for example, the authors considered Gamma and Weibull in addition to the already mentioned exponential, log-normal and power law distributions. We cannot provide any recommendation on the choice of distributions to be added in this case. However, the availability of parameter estimators and/or of the distributions themselves in the statistical analysis package can prove to be in practice a really efficient criterion. One good example here is the exponential+power-law compound distribution, which was shown in [46] to be a very good candidate to solve the dilemma between exponential and powerlaw for aggregated ICTs, but which is as far as we know not readily available in R. Parameter estimators for this precise distribution are also non trivial.

use the candidate which is able to capture the highest number of pairs for all pairs. This is the approach we used in Section 6.5 following [83]. Considering a single distribution has the advantage of simplicity, but the results in [83, 23] and in this chapter showed that real traces exhibit in fact more diversity.

take them as-is. Although some care definitely needs to be applied when performing a time and/or node number scaling, this approach has the benefit of retaining the biggest amount of information. Indeed, it involves very little processing.

discard them. This is likely to be the worst approach, since the data and behavior represented by these pairs will be lost for further processing. It will also reduce the total number of pairs in the experiment, and the degree for the corresponding nodes, which in turn will reduce contact opportunities and message transfers. Impact of these changes on network performance will typically be noticeable: for these reasons, we do not recommend it. Note that in this case, we end up with deleted pairs, which were discussed in the previous paragraph.

Another valuable question here is *Why* are these pairs unfitted? This may also help answering the previous question. For example, in Section 6.5, we chose to consider all pairs having at least two intercontacts (or three contacts, which is equivalent here), to leave as few pairs behind as possible. However, it is quite hard to get meaningful statistical results with so little data. Authors of [23, 87] chose higher contact numbers.

Is the tting method appropriate? We mentioned in Chapter 5 that CT and ICT data is made of discrete values, since the precision found in real traces never goes below 1 second. In addition, these values are sometimes repeated, thus producing ties. This fact is made even worse by the sampling processed used in data collection, as shown in Figure 6.1. Yet, the tools and distributions used in existing literature (and consequently, in our works as well) are all designed for continuous data. This is clearly

an issue from a mathematical standpoint. The problem also appears when preparing the synthetic trace, because the data generated by continuous distributions is continuous and requires subsequent rounding to achieve the same precision as the original trace (1 second in our case). Interestingly, we found out that with the log-normal distribution, some parameter combinations provided us with 0-second contacts due to the rounding operation. Furthermore, considering separate CT and ICT drops all information on simultaneous contacts, which were shown in [63] to be in fact extremely common, and which can also dramatically improve connectivity and performance. This is of course only possible if the node can handle multiple connections simultaneously. The scope to fit is also important. We know that simply considering the aggregate distributions as in [16] is bad, because it hides many details. Pairwise distributions are better, but still not sufficient according to the findings in [63]. Hence, deciding on which characteristics to capture remains an open question, out of the scope of this work. In [84], the authors try to produce a list of characteristics. They consider pairwise CT and ICT distributions and number, social structures, spatial preferences and periodic reapperances. But even with all these characteristics, discrepancies in network performance remain between synthetic and real traces. Also, the aspect of periodic reapperances, although it is key to common human behaviors, is not supported by statistical distributions.

How to perform the comparison between distributions? In the statistical analysis of traces, the aim is to find the ideal candidate to model the data, which means finding the statistical distribution which best matches the observed data. This requires a tool to perform the comparison between all candidates considered. We already mentioned that there was no consensus in the literature, and several tools were used for this purpose, such as the KS test or the Cramer-von Mises test. Determining which one should be favored over the other is for us an open question, although a pragmatic choice can be to select the one readily available in the statistical library used, if it does not offer both. This is also related to the previous question regarding the nature of the distributions chosen, because some tools are designed for continuous distributions, and other for discrete ones, as already mentioned in Chapter 5.

How to validate scaling approaches when not doing network performance assessment? The main issue of the present work is to be able to provide a tool for conducting network and application performance assessment in the case of DTNs. However, as was mentioned in Section 6.2.2, this is only one of the three use cases for real traces. While we did not really consider the two others in our work, they can be interesting to others. Indeed, it is unclear that a scaling approach which correctly captures ICTs and CTs will also perform well with social structures (communities, degree, bridging nodes...).

Are beacon-based traces representative of real apps performance? This is perhaps the most interesting question of the list. Indeed, our final goal in this Ph.D. work is to be able to predict the behavior of real mobile applications, without necessarily

having to resort to a field trial. Hence, the material used should be able to provide a good approximation of the reality if we want our predictions to be useful to application developers. Known issues of contact traces are the fact that they do not involve actual data transfer, or do not record connection quality [73]. The sampling period can also be longer than needed, thus not offering sufficient precision. Another possible representativity issue of traces comes from the fact that in a given experiment setup, only part of the nodes are instrumented. For example, the Rollernet experiment was conducted with 62 nodes in a crowd of 2500. The same observations would apply to several other datasets, including MIT Reality or Infocom '05. However, there is no proof that this subset of nodes is representative of the whole crowd. In [69], the authors show that to correctly capture the aggregated ICT distribution of the network, up to 75% of the nodes may have to be collecting information (depending on the quantity of data collected by each node). Unfortunately, the work did not cover pairwise distributions.

6.8 Conclusion

Contact traces collected during field trials are the most straightforward way for researchers to introduce some realism in their work. They are mainly used in the context of DTNs for social inference, performance assessment or statistical analysis. In this chapter, we show that the conditions captured by traces can be quite far from reality, due to the high number of other factors which can in uence the collection process (sampling period, device characteristics...). We also listed filters which have been applied to traces in various research works, and proposed recommendations on their use depending on the envisioned experiments. For all the filters, researchers need to be aware of their existence, especially when these ones are already incorporated in the files available in public repositories. Finally, two existing approaches aiming to extend datasets while retaining their intrinsic characteristics are presented. We found out that there are actually several hidden limitations (such as modeling all node pairs with the same distribution, or validating with aggregated measurements) which may prevent them to truly achieve their goal. Validation procedures also seem to be questionable.

Because of the large number of factors which have to be considered when using real traces for performance assessment, we have come to the conclusion that synthetic traces (i.e., contact traces produced with synthetic mobility models) should be considered with more attention. Unlike real datasets, synthetic models have the advantage of offering full control over their parameters. However, measuring their conformance to reality is still a challenge, considering that this is often done by comparing them with real traces.

We now conclude this first part of the thesis, where we dealt with the possible inputs for a DTN emulation system. First, we considered in Chapter 4 the possibility of deriving an analytical performance model for the drop ratio. This model can help us to predict the average drop ratio in a network made of sources, relays and destinations, in the case of one-packet node buffers. Alternatively, it can also help us to derive the maximum number of sources that the network can accept to keep the drop ratio bounded. We

derived the model in the case of an homogeneous network, and an heterogeneous one with two classes of nodes. Possible extensions were also presented. However, and like other analytical performance models, many simplifying assumptions are required to make the model mathematically tractable. This highly reduces the possible use cases for our model.

In a second time, we moved to real traces, which are a popular material in the literature when it comes to adding some realistic input to a study. Inspired by existing scaling approaches (CTG [16] and EMO [83]), we first showed in Chapter 5 that the filtering operations performed before conducting a statistical analysis are in fact far from being innocuous. We also showed that the in uence on network performance can also be noticeable, and is not necessarily linked with the in uence on statistical analyses. Then, we extended in Chapter 6 this work on filters to the whole process, from collection to scaling, and tried to list all the filters and factors which can play a role, as well as the limitations of existing approaches on some examples. This work allowed us to provide a series of recommendations for future studies. We also gave an overview of open questions which we encountered during the course of the study.

Part II

The HINT Emulation System

Chapter 7

HINT: HINT Is Not a Testbed

Contents

In the previous part of the thesis, we presented two possible ways of producing input to an emulation system: analytical modeling and real contact traces. Through our own experience and developments, we showed the limitations of both approaches. In this second part, we move forward to the emulation system itself. More precisely, we first describe the challenges such a system would need to solve, and the associated requirements. Then, we are able to propose HINT, a new hybrid DTN emulation system, comprising both real nodes (as Android smartphones) and simulated ones. In this chapter, we also provide details on the implementation of our proposal. Going back to Table 2.4, we aim to bring down the cost of emulation while also improving the ease of use.

The rest of the chapter is organized as follows. In Section 7.1, we present the challenges of existing network characterization approaches which motivate the development of an emulator. Then, Section 7.2 lists the requirements derived from these challenges. The high level architecture of HINT is introduced in Section 7.3, while Section 7.4 details the corresponding implementation and technical choices. We discuss it in Section 7.5,

and conclude in Section 7.6.

Note: part of this work was published in ACM CHANTS 2016 under the title *HINT: from Network Characterization to Opportunistic Applications*. http://dx.doi.org/10.1145/2979683.2979694

7.1 Introduction

Emulators can operate at different layers of the network stack. For simplicity, we will target layer 3 (network). Consequently, effects to be replayed include connections and disconnections dictated by CTs and ICTs, packet losses and bandwidth changes. Behaviors at lower layers are abstracted.

Based on the existing literature, we can list the components that are required for our emulator:

real nodes, running the application to be tested;

simulated nodes, either individually or as an aggregate. Due to the impossibility to successfully aggregate the network and achieve our single-node model mentioned in Chapter 2, we will simulate each node separately instead of the network as a whole.

a layer to connect real applications to the emulator. It can be a working DTN stack implementation like IBR-DTN [75], or a custom interface made from scratch.

connections between nodes, either wireless or wired, allowing real *and* simulated nodes to communicate together;

a way to apply the connection impairments to the actual network links (disconnections, bandwidth limitations...);

a management system to drive the experiment, perform changes and collect results from the different nodes.

To the best of our knowledge, this is the first time that fully real nodes (operating system, hardware, software) in the wild are part of a lightweight DTN emulation system. Furthermore, none of the proposals introduced in Chapter 2 are designed with application development in mind.

In this work, we want to study opportunistic networking. For this paradigm to become a reality, it will be necessary to make use of connected devices which can be easily carried by users at any time. Current DTN testbeds mostly use computers in their setups, but laptops are too bulky to accompany users during their daily routines. For this reason, we choose to rely on smartphones to act as the real nodes in our emulation setup.

7.1.1 Current challenges

Existing approaches to assess the performance of a DTN have already been introduced in Chapter 2. Analytical models were presented in Section 2.2. One of their major issues is their impossibility to change scale or modify any of the assumptions required for their derivation. Furthermore, no model can simultaneously provide all the metrics which would be of interest to an application developer, and even when they do, they may only provide the average value instead of the complete distribution. This drawback is not present in simulators, like the ONE [47]. However, simulators are not designed to handle real applications because they mostly focus on routing. For completeness, we should note here that the ONE offers support for the DTN2 stack [25].

Trace collection aims to capture the intrinsic mobility characteristics of an actual network setting through contacts and intercontacts, and abstract the underlying aspects like the link layer. Unfortunately, Chapters 5 and 6 showed that this abstraction was much thinner than expected. Indeed, traces are in fact affected not only by the link layer, but also by other assumptions and choices made by the experimenters at various stages in the process. This makes traces less representative than needed, and therefore less useful to application developers.

Being able to access network characterization and properties is necessary for application development, due to the complexities of opportunistic networks. For example, metrics like delivery ratio or delay cannot be neglected. Failure to obtain information about the network would require making hypotheses on its characteristics, which often leads to over provisioning and resource waste. In the already challenging context of opportunistic networking, a possible waste of resources is a major issue that is best avoided. However, developers may not have the adequate knowledge or resources to correctly exploit network characterization.

Furthermore, network characterization is not sufficient from a developer point of view: the impact of the network properties on the application and the resulting user experience also has to be taken into account. Questions to be answered here are: *Is my application working as expected?* or *Is my application still working in an opportunistic use case?* This requires the ability to insert real applications into the test system. For this purpose, real deployment is the candidate which offers the highest degree of realism, but also the highest setup cost, especially in an opportunistic use case where mobility and connection impairments are frequent. Consequently, we have to turn to emulation. Testbeds are great when it comes to realism, but suffer from a high deployment cost which makes them unsuitable for integration within a development work ow. Ideally, we need to offer an emulation system which offers sufficient realism to derive results meaningful to developers, while at the same time being lightweight enough to fit into existing development environments.

7.2 Requirements

The study of existing alternatives conducted above allows us to provide a list of requirements for the system. They can be divided in two groups: link layer requirements and emulation requirements. We also explain the difference between connection-oriented and contact-oriented emulation.

7.2.1 Link layer requirements

One problem of opportunistic networks is the lack of an adapted communication technology which could be used by real applications. Indeed, existing link layer technologies such as Bluetooth or Wifi Direct lack many of the assumptions made by opportunistic network algorithms. Here, we define two requirements which we believe should be met by any communication technology supporting opportunistic applications.

C1 **Connection bi-directionality:** algorithms proposed over opportunistic networks can assume connections to be bi-directional. See for example epidemic routing [88], where the first step is the exchange of a summary vector giving a list of all messages: such an exchange is impossible in the case of a unidirectional connection.

C2 **Multicast communication:** nodes can connect with multiple devices at the same instant of time over opportunistic networks. This is also an underlying assumption in [63]: k-intercontacts are impossible if each node can only be connected to one neighbor at a time.

7.2.2 Connection-oriented vs Contact-oriented emulation

Simulators are usually implemented in a connection-oriented basis: each time a contact between two nodes occurs, a global state is set to denote an open connection between nodes. As long as the connection is open, the simulator will try to send and receive data. This open state is updated at each time step of the simulation until a disconnection occurs. At this moment, any non-finished transfer will be dropped.

Instead, we propose for our emulator a contact-oriented approach: since we know the duration of the contacts beforehand, we can pre-calculate which messages may be actually transferred within the duration of the connection. This allows us to optimize the number of events to be generated while keeping the same behavior, instead of updating the state of the connection at each step of the emulation. Furthermore, this matches more closely the event-driven approach of our emulator.

7.2.3 Opportunistic emulation requirements

The goal of our emulation system is to allow developers to test their applications. Generally, developers interact with IDE tools for all activities related to application building, such as testing, coding or debugging. This means that our system will also have to integrate in such work ows. We now define the following requirements for our emulator.

E1 **Real-time emulation:** To be able to test an application, the test system should run at the same speed as it. Consequently, our emulator must be real-time. Notice that this differs from a simulator or a testbed in the sense that we focus on the application development process and not on the network characterization.

E2 **Contact-based emulation:** In order to simplify the view of the emulator, we think that a contact-based emulator must be put in place. Contacts are easy to understand in terms of behavior, while abstracting the complexity of node mobility. Since we do not rely on a particular link layer technology to deploy opportunistic applications and based on the requirements (C1) and (C2), we can abstract the network and still conceive fully functional applications.

E3 **Real-time tuning:** Because the parameters of opportunistic algorithms can be cumbersome, we argue that a real-time parametrization must be put in place to better understand the impact of changes in the emulated opportunistic networks. This allows developers to better plan for changes and adapt their applications accordingly. Such a feature is not supported by simulators.

E4 **Real-time monitoring:** In order to tune parameters and assess the impact of changes, the need to observe the behavior of the network is very important. Furthermore, these observations have to be made in real-time, without having to wait for the end of the experiment as for a simulator. For that, we argue that any opportunistic network emulator needs a real-time monitoring system.

E5 **Transparency:** The application that is being tested should not be aware that it runs over an emulator instead of a real network. This also means that we should, ideally, be able to run the software with as few modifications as possible. Finally, if we want the results to be useful to developers, it is crucial to ensure that the emulation platform itself does not bias application operation.

E6 **Repeatability:** For the debugging of an application, it is crucial to be able to exactly reproduce the conditions for an error to happen. Consequently, an opportunistic emulator needs to support exact reproduction of simulated network conditions over several distinct runs if necessary for the end user. This repeatability is also highly desirable in order to debug the emulator itself.

E7 **Availability:** For a greater adoption, the emulator should be easy to install and run. Unlike a testbed, it should ideally be able to run on more limited hardware resources like a single computer, just like other IDE tools (think about the Android emulator). It should also be easy to access, without requiring complex administrative procedures such as booking time slots in advance. Although fully understandable for testbed management purposes, such procedures would quickly be inconvenient for a development process, where testing needs are hard to predict in advance.

7.3 General architecture

In order to fulfill the requirements expressed in Section 7.2, we propose HINT. HINT stands for *HINT Is Not a Testbed*. Recall that our emulation system aims to be a tool for developers, which should therefore be lightweight enough to fit into existing development environments. This is the meaning of requirement (E7). For this reason, HINT does not involve any virtual machines. A second feature of our emulation proposal is its ability to provide the developers with *hints* on how their application would behave when running on an opportunistic network, without having to do a real world trial. In this section, we present the architecture of our emulator, first at a high level, then by detailing each of the five building blocks of our proposal.

7.3.1 High-level architecture

There are two interaction levels in our architecture: the real world and the emulated world. In the real world, real devices (physical Android mobile phones, in the present case) run the application to be tested and the emulator service. In the emulated world, virtual nodes and real nodes interact in an opportunistic way in a unique network.

Our emulator creates and manages virtual nodes according to user requirements. Therefore, the resulting emulated opportunistic network is composed of several opportunistic nodes. The emulator also defines the connections between opportunistic nodes at the emulated level, and applies changes in real time according to contact opportunities. Note that real nodes can only communicate with the emulator (at the real world level), and not directly with each other. Hence, we ensure that all connections go through the emulator. This is absolutely necessary, because in our emulation platform, real nodes are going to be quite close to each other in reality. This would mean that they could be in permanent contact with the other real nodes, and may also interfere with each other. These two features are undesired for an emulation system, and would contradict requirement (E5) on transparency. Several network topologies can be drawn by changing connections, according to the user scenario.

Figure 7.1 shows both interaction levels with the projection of the different opportunistic nodes on the emulation plane. Solid lines indicate that two opportunistic nodes are in range (contact opportunity), while dashed lines represent real connections (a mobile phone connected to our local network hosting the emulator through Wi-Fi).

7.3.2 System architecture

We can now introduce the general architecture of HINT. It is made of 5 building blocks, presented below and in Figure 7.2.

Core Emulator

It is a real-time event-driven system, in charge of running the emulation scenario. A scenario specifies the following parameters: number of real and virtual devices, node characteristics, contacts and intercontacts management, message creation patterns, message

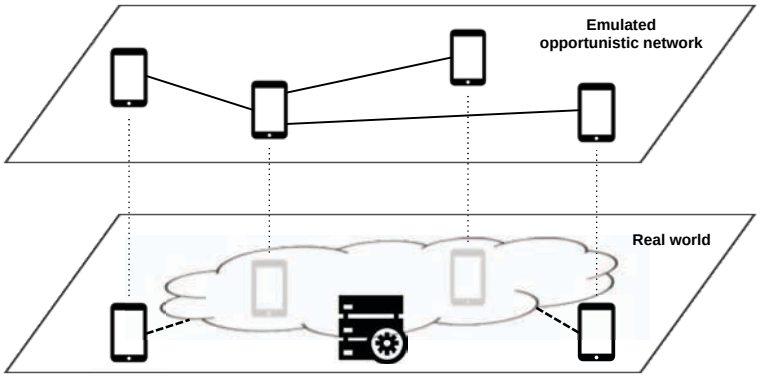


Figure 7.1: High-level architecture, showing both interaction levels: real world with real devices, and emulated world with opportunistic (real or virtual) nodes. Solid lines indicate emulated connections, while dashed lines denote real connections with the emulator.

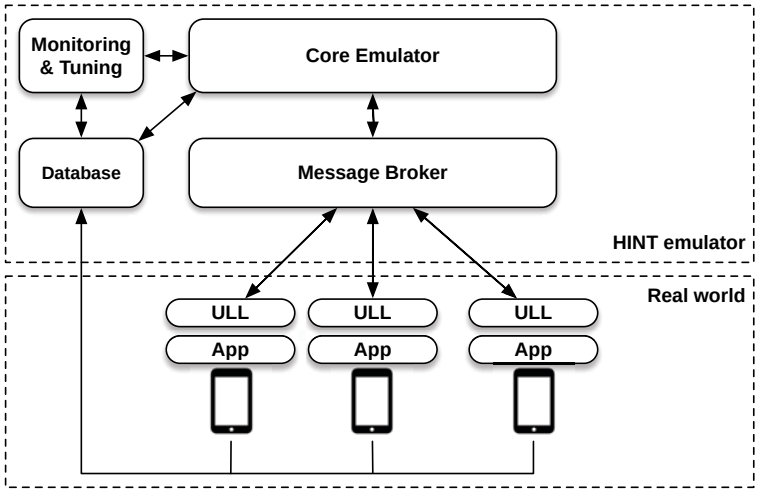


Figure 7.2: General architecture of HINT, with the 5 building blocks.

forwarding and routing strategy, buffer management. Events, such as contact durations, intercontact durations, message creation and forwarding, are created and scheduled for execution in an event queue. Routing decisions and buffer management are also handled for all nodes, real and virtual. Although the functionalities listed above are very similar to those offered by a simulator like the ONE [47], the Core Emulator supports in addition real messages (and their corresponding payloads) and real-time operation.

Message Broker

This module manages the interactions between all nodes. We use the Message Broker in order to store both system messages and node messages. Indeed, events from the Core Emulator generate meta-messages (i.e., announce connection start and duration) that are sent to the broker to a system queue. Regular messages contain the actual application data exchanged through the network and are stored in dedicated queues for each node. More precisely, three queues are defined: an Inbox queue for incoming messages, an Outbox queue for messages requiring further routing and a Storage queue for received messages. Hence, the Message Broker allows the communication between each pair of nodes (real or virtual). Since the number of pairs can be large, a specialized service that can scale is needed.

User Link Layer

It abstracts the communication between the emulator and the real application, thus making the application unaware of the emulator. It communicates messages from the node queues to the real application, or from the application to the node queues. This layer can also be used to plug-in different DTN stack implementations. In some way, it plays roughly the same role as a DTN stack. This part is embedded in real nodes.

Database

This module interacts with the Core Emulator to manage all nodes. It stores the characteristics of each node: device type and ID, connection parameters, message creation frequency, etc. It also contains data for the monitoring part, in order to avoid unnecessary polling of the Message Broker queues.

Cross-layer Monitoring and Tuning

It is a cross-layer module where statistics are collected and displayed in real-time. Three views are offered: full network, node pairs or single nodes. This layer also allows to change parameters such as node buffer size or link speed. Unlike a simulator, monitoring and tuning can both be done during the experiment, instead of having to wait for the end of the experiment to plot graphs. For simplicity, we support the use of the same tuning interface to specify the initial emulation scenario in addition to Python-based scenarios as explained later.

These building blocks are shared between the different devices: each real device runs the User Link Layer and the real application being tested (as implied by Figure 7.2), while a central computer runs the Core Emulator, the Message Broker, the Monitoring and Tuning system and the Database. Note that thanks to its web-based nature, the monitoring and tuning can either be done from the central computer, or from any other computer able to access the central one over the network. Furthermore, the Message Broker and Database can also run on machines distinct from the one running the Core Emulator.

We can also check that the architecture presented above complies with the requirements introduced in Section 7.2. The Core Emulator is a real-time system (E1). It produces a contact-based emulation (E2). The Cross-layer Monitoring and Tuning part addresses requirements (E3) and (E4), while the User Link Layer abstracts the communications with the emulator, allowing any application to run on top of it. This achieves (E5). Finally, the Core Emulator is able to replay a scenario with the same contacts, routing and message generation parameters, which realizes (E6). To show the availability (E7) of our architecture, we can note that it only has limited hardware requirements: smartphones as real nodes, a computer to run the emulator and a WiFi access point to connect all devices to the same network, as we will see shortly. This is much lighter than a full testbed, and consequently more adequate for integration in a development environment.

Finally, we can compare the contents of the building blocks presented above with the list of components of Section 7.1. In HINT, we can find both real and virtual nodes. The User Link Layer binds the real nodes with the emulator. Connections, disconnections and impairments are managed by the Core Emulator. Results collection is also made by the Core Emulator, while tuning and metrics display are handled by the Monitoring and Tuning system.

7.4 Implementation details

In this section, we provide a precise description of all the components involved in HINT. This description is based on the list given in Section 7.3, but also includes the other utilities and parts which are required by HINT. We developed this prototype in Python 3.5 with the RabbitMQ message broker and MongoDB database.

An overview of the physical system is provided in Figure 7.3. In this example, the central computer is a MacBook Pro with 16 GB of RAM and a 2.5 GHz Intel Core i7 processor. Real devices are Nexus 5X with 32GB memory, running Android 6.0.1. A dedicated WiFi a/g access point can also be seen to include all physical devices and the emulator in the same network. As mentioned in the previous subsection, these hardware requirements are sufficient to experiment with HINT.

Figure 7.3: View of the actual system, made of a central computer, a WiFi access point and two real devices as Nexus 5X smartphones running the opportunistic application being tested.

Core Emulator

The Core Emulator defines an emulation scenario and all its parameters using the Scenario class. The world contains Nodes. Node interactions are modeled as events. More precisely, they can be placed in 3 categories. First, we have the usual Contact and Intercontact events, required by our contact-oriented approach. Then, to perform routing, we define Creation, Copy, Forward, Delivery, Drop and Expiry events. These events are also especially useful for triggering database updates for the monitoring system. Contact, Intercontact and Create messages are generated in batches, and we also define Refill events as system messages to trigger the creation of new events when batches are almost consumed. This regeneration happens on a pairwise basis. Finally, the Tuning system creates Tuning events to propagate the changes requested by the user through the web interface.

We use a priority task scheduler to handle all the events queued. Each event keeps information about the pair of nodes involved and its execution time. Events are executed in real time accordingly to the execution time. Contacts and intercontacts are generated according to user inputs, and can come from real traces or statistical distributions as needed. The number of virtual and real nodes is currently fixed for the whole experiment length; a variable number is left as future work.

Currently, the Core Emulator only implements a limited subset of routing and buffer management policies. We support Direct Transmission, which means that a packet will only be transmitted to the destination; Always Forward, when all messages are forwarded at each contact; 2-hop routing; and Flooding, when all messages are copied at

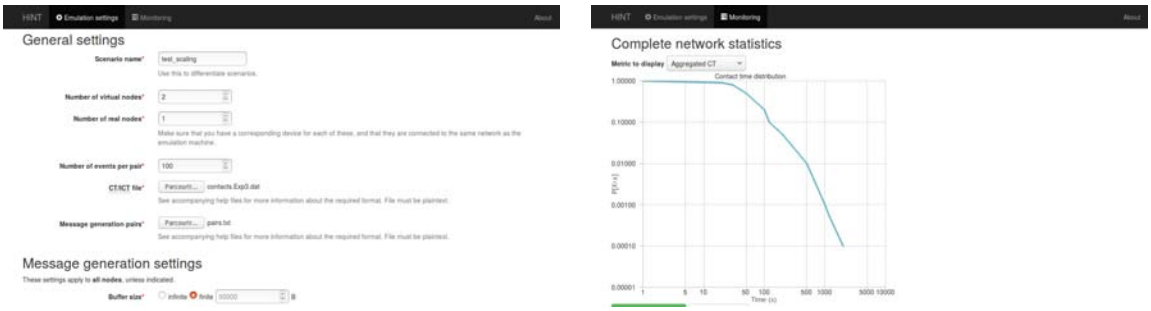
each contact (similarly to Epidemic, but without the summary vector exchange). For buffer management, we implement drop_oldest (aka *drophead* in [97]) and drop_newest, along with a policy which stops accepting packets once the buffer is full (called *droptail* in [97]). Following [60] and [97], these three approaches have a complementary implementation which performs the same, except that it tries to avoid deleting packets created by the node when drops are required. They are referred to as Avoid Source Packets versions of the policies. In addition, all buffer management policies have an optional support for TTL. Note however that the implementation we propose is designed to easily accommodate new routing protocols or buffer management policies: they are indeed implemented as classes, with each Node having its own router and buffer management policy instance. The rationale behind this choice is to be able to accept routing protocols keeping additional information (on encountered nodes or messages already seen, for example) which can be different for each node. Other possible scenarios could involve mixing different router types within the same network, or a routing protocol change in the middle of an experiment.

Message Broker

We use RabbitMQ, taking advantage of its high scalability to enable the execution of all events in real time with an increasing number of nodes. We define two queue types. Meta-messages (contacts, intercontacts) are stored in the *system queue*, while *node queues* represent node buffers, both for real and virtual nodes. Each node has three queues: Inbox, Outbox and Storage. When a message is sent, it gets forwarded (or copied, depending on the routing protocol used) from the source Outbox queue to the destination Inbox queue. This first step always succeeds. If the destination is not the final recipient for the message, it will then try to move the message from its Inbox to its Outbox, taking into account any message expirations or drops (caused by a finite buffer size) which may be necessary at this stage. If the destination is the final recipient, the message will be added to the Storage queue and will no longer contribute to the recorded buffer occupancy for this node. Messages created by the application running on the real nodes are placed in the Inbox queues corresponding to the source node, and undergo further processing by the Core like the other messages. Note that to make debugging easier, the size of the messages does not necessarily re ect the actual payload size: for example, sending a message of a size of 1 MB with an empty payload is allowed in HINT.

Database

We use MongoDB. We store node characteristics, such as their connection parameters, buffer size, current list of neighbors or degree. We also store all Message events (creation, copy, drop, delivery, but also contacts and intercontacts) which are required to compute and display network metrics in the Monitoring system. Note that this stage does not require the storage of the actual message payload, but only metadata. Finally, general data on the emulation scenario (number of nodes, message generation, contact and intercontact parameters) is also recorded.



(a) Scenario edition view (b) Network CT distribution view

Figure 7.4: Example screenshots of the monitoring and tuning system.

Cross-layer Monitoring and Tuning

We implemented a simple web-based interface to present node, network and pair statistics for real-time monitoring. We are able to present the evolution over time of DTN performance metrics such as message delay, delivery ratio, hop number or contact time distribution, to name a few. Indeed, these metrics will be required to validate the emulator results against other approaches like simulators.

This web system also allows us to tune parameters during the experiment both at the network or node level, such as node buffer size or connection speed. Monitoring and tuning is conducted through a REST API, with graph data being sent as JSON. More precisely, the monitoring URLs for network, node and pair are defined respectively as `/m/network/<metric>`, `/m/<id>` and `/m/<id1>/<id2>/<metric>`. For the tuning, URLs are defined as `/t/network` to change network parameters and `/t/<id>/<parameter>` for node parameters. Screenshots of the web interface are presented in Figure 7.4. It is implemented using jQuery, jQplot and HTML5 with Bootstrap, which allows us to provide graphs in addition to an access to the raw data. Evaluation of the application behavior with respect to user experience is done directly on the real nodes.

We tried to cover the major network performance metrics that a DTN researcher may be interested in. However, it may be necessary to add some in order to study other properties. Doing so would require to interface directly with the MongoDB database if the information is already available as a Node parameter or in the message metadata. Otherwise, deeper modifications of the emulator would be needed, mostly in the Core (for data recording) and Monitoring (for graph plotting) parts.

Contact and intercontact input

The aim of HINT is to allow researchers to use any type of input for their experiments: real traces such as those studied in Part I, synthetic mobility models outputs or distribution-based traces. In order to accommodate the latter, we designed Henget. The purpose of this utility script, written in Python, is to let researchers generate traces based on statistical directives. Considering the advice given in Section 6.6, we support

three directive types:

homogeneous distribution, when all pairs follow the same distribution with the same parameters (see for example the homogeneous case of Chapter 4);

EMO-style , which follows the approach detailed in [83] of a single distribution for all pairs, but with different parameters for each pair, these parameters being themselves generated by a given distribution;

full heterogeneity, where each pair can have its own distribution and own set of parameters for it. Although we are not aware of any tool supporting such a degree of heterogeneity at the moment, we hope that this situation will change. This mode is also the only one to support pairs of nodes with no contacts at all, which almost always exist in real traces. Note that to generate a trace with different parameters for different groups of nodes as with the model of Chapter 4, this is also the operation mode to use.

The Henget utility relies on the scipy scientific library for statistical distributions. At the moment, the script itself only supports exponential, Pareto and log-normal distributions, which were already used in the literature or in Chapters 5 and 6. We also added for convenience a dummy distribution which always returns the same value. If needed, other distributions, such as Gamma or Weibull as in [83], can be easily added from scipy.

This versatility comes at a cost: indeed, Henget relies on inputs as text files to properly generate the requested data. Each of the three directive types expressed above requires a different input format for text files. To support instantaneous contacts, only intercontact generation directives have been made mandatory. In this case, all contacts are set to a duration of 0 second. Note that with this duration, neither HINT nor the ONE [47] will be able to route any packet.

To produce other traces, the ONE [47] and BonnMotion [3] allow to generate and export contact traces based on several synthetic mobility models, including but not limited to those surveyed in [17]. For real traces, applying the recommendations introduced in Chapter 6 on material found on CRAWDAD can be an effective solution. Note that the ability to use traces generated outside the Core Emulator enhances the repeatability (E6) of the system, and is also beneficial for debugging purposes. At this point, we would like to remind the reader that one of the main requirements of HINT is the symmetry of contacts (C1), a symmetry which is rarely assumed in real traces. Consequently, some preprocessing would be necessary before using such traces with HINT.

Going back to the Core Emulator part, which actually performs trace reading, we chose to use an event scheduler using a relative timestamp instead of an absolute one, with the 0 being the start of the internal event loop. The consequence of this choice is that users of real traces will have to modify the time reference of their datasets. This operation can be achieved very easily with an awk or Python script. The interested reader may want to have a look at the Git repository of Adyton [60][1], where examples of such scripts for several real traces can be found.

[1] https://github.com/npapanik/Adyton

Apart from the time reference and symmetry changes, HINT has been designed to handle natively the format used by Rollernet [5] and Infocom 2005 [76]. The trace format used by the ONE [47] is not supported yet.

User Link Layer

In most of the existing proposals (see for example [8, 50]), emulator nodes are connected to two separated networks. The experimental network is used to carry application traffic and receives the impairments required, while the control network is left untouched and only carries management and monitoring data. The separation between the two is required to ensure the control network does not make any interference with the experiment, thus ensuring transparency of the emulator and validity of the results. It was however shown in [52] that traffic on the control network was almost negligible compared to the amount of experimental data transferred.

In our case, the situation is slightly different, because our nodes are smartphones instead of computers. This means that we need to work with the existing network interfaces, and cannot modify hardware specifications for our purposes. On a typical smartphone, we can rely on Wifi, Bluetooth and cellular data (3G/4G). If we want to replicate the dual network system of existing proposals, a choice is needed. Relying on cellular data is not practical, because it requires subscriptions for all the real nodes and Internet access for the Core Emulator. Furthermore, cellular network connectivity is highly unlikely to be used in a DTN context. This makes cellular data unsuitable for both networks. Then, we have Bluetooth. This technology was used in the real traces considered in this thesis work (see Section 2.3), which makes it at first glance a good fit for DTN studies. However, this would require a Bluetooth adapter on the machine hosting the Core Emulator, and limit the amount of real nodes we can use to values supported by the protocol, which are rather small. Furthermore, the study of traces conducted in Chapter 6 showed several limitations of Bluetooth related to missed contacts or synchronization problems. The risk here is to add impairments because of the emulator and not because of the requested emulation settings. This would contradict the goal of Transparency (E5).

Consequently, we choose to use Wifi. Now that we have only one connection left, it follows that both control and experiment data will have to go through it. The need for control data is the reason why we use an always connected link: if we actually got a disconnection, it would then be impossible to restore it (i.e., recreate a contact) without manual intervention. Also note that at the time of this writing, IBR-DTN [75] only supports Wifi and Wifi Direct, which is another strong argument in favor of choosing Wifi.

For convenience, we implemented the User Link Layer as an Android service running on the real devices. No DTN stack is currently supported: this point will be covered in Section 7.5.

To allow any application to interact with HINT, the User Link Layer exposes an interface to seamlessly send and receive messages to and from other nodes through the HINT emulator core. Indeed, to send a message the application simply needs to call

the appropriate User Link Layer method. To receive messages, the application declares a standard Broadcast Receiver subclass that appropriately filters and handles incoming messages. When messages are forwarded to a node within the HINT emulator core, they are placed into Message Broker queues. Then, the broker fires a notification to the User Link Layer of the corresponding real node. Finally, the message is consumed from the broker by the User Link Layer and passed to the application by means of a Broadcast Intent to be finally used by the real application.

From a developer perspective, using the User Link Layer in the application means importing a package, and designing a view to adjust settings, such as the one presented in Figure 7.5. It also means providing methods to communicate with the service.

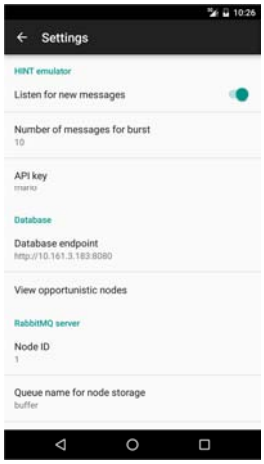


Figure 7.5: Screenshot of the Android application part representing the User Link Layer, with some of the required settings.

Real application

Finally, we propose a simple message application developed for Android. This application allows to receive and send messages to other peers of the network through the User Link Layer. Because routing and buffer management is made by the Core Emulator, the User Link Layer only has to perform message exchange to and from the emulator, and notify the application when a message is available as explained above. A preview of the main application window, called OppChat, is provided in Figure 7.6. We will come back to it in Chapter 8.

7.5 Discussion

We use Python because of the simplicity it offers for development and prototyping. Should this create performance issues, a reimplementation in another language like C++ would be necessary.

Figure 7.6: Screenshot of the Android chat application OppChat developed for the project. (Mario never came to the restaurant, he was too busy saving Princess Peach.)

The three-layered architecture provides a good modularity with separation of concerns (Core Emulator, Message Broker and User Link Layer). This helps to keep track of the big number of events and messages of the emulated network, thus ensuring scalability. Also note that the components themselves run on different machines: application and User Link Layer run on the real nodes, while the other modules are found in the emulation machine. If necessary, the other modules found in the Core can be split over several computers, although this would make HINT less convenient to deploy.

The Core Emulator can be extended with more routing algorithms or buffer management policies if needed, thanks to our implementation choices based on class inheritance and instantiation of one router or buffer manager per node. The use of a message broker helps to multiplex events and provides an original way to implement node buffers. Also, several message queues (three per each opportunistic node) are used and allow to provide node metrics such as buffer occupancy. The User Link Layer plays the same role as DTN stacks, such as IBR-DTN [75], ION [11] or DTN2 [25]. Finally, we provide a real-time execution emulator with a much lower hardware and financial cost compared to a testbed or a real world deployment.

Concerning real nodes, one may wonder why we chose to only leave an abstraction layer (the User Link Layer) on the real nodes, and put everything else (routing logic, buffer data) in the Core Emulator. Indeed, in the actual DTN stack IBR-DTN [75], each real node embeds a message queue and routing logic for 5 different protocols. We made this choice to avoid replicating code (and the corresponding logic) in two different places, namely the Core Emulator for virtual nodes and the User Link Layer for real nodes. This option is made possible thanks to the centralized nature of our emulator. For fully decentralized systems, like a real DTN, this option cannot be offered, thus justifying the design chosen for IBR-DTN.

Then, an experimenter may want to replace the User Link Layer we provide by a

DTN stack, for example to get closer to standards compliance. For Android, the only choice at the time of this writing is IBR-DTN. At first glance, looking at the high level architecture depicted in Figure 7.2, this replacement should be the only change needed. However, using an already existing DTN stack in HINT would create an issue with our centralized architecture when it comes to routing: indeed, the DTN stack would require information about connections and disconnections, which are currently restricted to the Core Emulator. During our preliminary experimentations with IBR-DTN, we noticed that contacts were detected using replies to a multicast discovery message, which was sent by all nodes to the same address. This approach well adapted to an actual DTN context would be highly problematic in our emulation system. Because the real nodes are typically all going to be connected to the same wireless access point, this means that they will all receive the discovery messages sent by the others, and consequently enjoy a permanent connection with each other. A possible workaround to this issue could be to deactivate the discovery mechanism, but this would still leave us with a need to recreate it somehow to trigger connections and disconnections from the Core Emulator. Such modifications of the Core Emulator (and presumably also of IBR-DTN) are left as future work. Also note that we would have to use a DTN stack for virtual nodes as well, which is currently not supported by our current architecture.

Finally, a major issue which arises with distributed systems, and DTNs in particular, is the need for a common time reference across devices [93]. This is for example mandatory for TTL expiration. In HINT, all routing, buffer management and drop decisions are managed in the Core Emulator, which runs on a single machine. This architectural choice allows us to remove the issue because with only one machine, no synchronization is needed. Again, the integration of an actual DTN stack doing its own timestamps using the device clocks would reintroduce the issue.

7.6 Conclusion

In this chapter, we moved from input preparation to the design of an emulation system. First, we argued that there is currently a gap between the results provided by opportunistic network characterization and the needs of application developers. Due to the inherent complexities of opportunistic networking, solving this dilemma is an absolute prerequisite for making opportunistic applications a reality. We argued that an emulator can be a valid solution.

First, we listed the limitations of current network characterization approaches, to derive a list of requirements for our emulation system proposal. The result, called HINT, is a lightweight hybrid emulation system specifically aimed at developers, comprising both real and virtual nodes. According to the classification of [7], it is a centralized low-complexity topology emulator, developed as a distributed real-time event-driven system. We proposed a high-level architecture for HINT, along with more details on the corresponding implementation in Python 3.5. Finally, we discussed some of the architectural choices we made for our emulator.

In the next chapter, HINT will be used for several experiments to validate its ability

to effectively fulfill the requirements set in Section 7.2. Then, we will conduct some real case studies of opportunistic application development with it.

Chapter 8

Using HINT

Contents

The architecture itself, although necessary, is not sufficient to build a successful emulation system. In this chapter, we use HINT to perform various experiments.

More precisely, when it comes to emulation systems, two steps are required. First, the proposal should be able to work and provide meaningful results. This is what we call the Validation stage, covered in Section 8.1. Three criteria can be checked at this stage: scalability (ability to handle a large number of nodes seamlessly), realism (ability to provide results close to other approaches like real experiments or simulations), and repeatability (ability to provide similar results in several runs). Existing literature mostly focuses on the first two items.

Then, once the system is known to work, the next step is to use it and perform the application tests and experiments it was originally designed for. Consequently, we present in Section 8.2 a case study with a custom-made DTN chat application, OppChat.

Note: part of this work was published in ACM CHANTS 2016 under the title *Demo: Using the HINT Network Emulator to Develop Opportunistic Applications*. http://dx.doi.org/10.1145/2979683.2979699

8.1 Validation

We now turn to the validation of the system. We choose to verify its scalability and realism. Generally speaking, *scalability* refers to the ability to achieve larger scales, which in the case of an emulation system like HINT can be defined in terms of nodes or of events. *Realism* means in the present context the ability to provide the user with meaningful results which would ideally mirror the ones obtained in a real setup.

8.1.1 Scalability: number of events

As a first step, we choose to validate HINT by assessing its capacity to scale to a large number of events per second. In this experiment, we are interested in how many contact (and intercontact) events our system can process each second while maintaining an acceptable delay. By doing so, we want to evaluate the ability of the emulator to handle real-time operation. No other messages are exchanged at the moment.

We only consider virtual nodes for this evaluation. Since all buffers and events are managed within the Core Emulator, independently of the node type, the absence of real nodes should not restrict the applicability of the results. The emulation machine where the Core Emulator resides is a MacBook Pro with 16 GB of RAM and a 2.5 GHz Intel Core i7 processor. We use contact and intercontact events with a fixed duration of 1 second. The experiment duration is set to 20 seconds to allow us to repeat it several times. Although such parameters are not realistic in an opportunistic network setup, this experiment aims to find out the limitations of our system rather than evaluate network performance.

The results are presented in Figure 8.1, which shows the ratio of delayed events (denoted as missed events) for different time thresholds ($T = 0\ 01$, $0\ 1$ and 1 s) and system loads ranging from 1 to 1225 events scheduled per second. This variation is achieved by varying the number of nodes in the experiment from 2 to 50, and assuming that all possible pairs play events (which is never the case in real traces). Recall from Chapter 7 that connections are bidirectional. We compute each ratio by performing the mean over 5 emulation runs. Note that we use a logarithmic scale for the x-axis.

From the chart, it can be seen that a time threshold of $T = 0\ 01$ s is not achievable with the current configuration of the emulator, because delayed events start to appear with loads as low as 3 events per second. As a result, for this threshold, the number of missed events increases exponentially as a function of the system load. On the contrary, the emulator can handle up to 10 events per second with a delay constraint of $T = 0\ 1$ s. Finally, with a time threshold of $T = 1$ s, our emulator is able to schedule and execute up to 300 events per second without additional delay. Although a threshold of 1 second can seem very high, especially for software claiming to achieve real-time operation, we have to mention that this value is the granularity used for real contact datasets.

Limitations come from the internal Python scheduler we use. Indeed, our current implementation never deletes events. So, when an event starts to experience a delay, the scheduler will fall behind and the delay will be propagated to the following events and accumulate. Unsurprisingly, the values also depend on the hardware specifications of

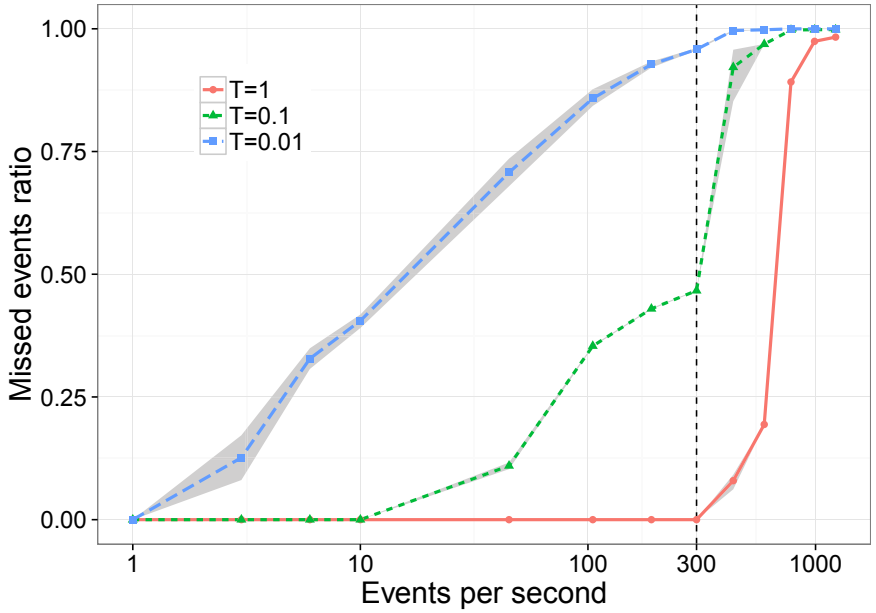


Figure 8.1: Percentage of missed events for different time thresholds (T, expressed in seconds) and system loads (events/s) (log-linear scale).

the host computer running the scheduler, as well as various other options of the Python interpreter.

One could argue that the number of events that HINT can handle per second is rather small, thus impeding the scalability requirement of our proposal. At this point, it could be interesting to compare the number of events that we are able to execute in one second with the ones experienced in real traces. For this analysis, we will go back to the four traces presented in Section 2.3, and provide for each one the average and maximum number of events per second (contacts or intercontacts). Rollernet [87] and Infocom 2005 [76] are considered in full, without any filtering. For Humanet [12], we consider the whole trace length, with all internal devices. The MIT dataset [28] is considered for 180 work days, only with internal devices, and considering only Bluetooth contacts. The aim of these choices is to leave as little data behind as possible, to derive meaningful upper bounds that HINT should be able to handle. The values are presented in Table 8.1.

Table 8.1: Average and maximum number of (inter-)contacts per second for four real traces.

Trace	Average	Maximum
Rollernet	26	146
Humanet	5.5	102
MIT 180 days	0.008	34
Infocom 2005	0.22	26

We should note that even on a very dense real trace like Rollernet, the maximum number of events recorded in a single second is 146. These figures are obtained on the raw, unfiltered data, comprising all possible contacts and all 1112 nodes observed during the experiment. Actual values of number of contacts per second are much smaller: for example, the average value on the Rollernet dataset is 26 events per second, which is the highest value of the four traces considered here. Also note that applying some of the filters described in Chapter 6 would reduce these figures even more, and that one specificity of the Rollernet trace is a very high percentage (75%) of 0-second contacts. This means that for most contacts, the following intercontact is scheduled on the same second, thus artificially increasing the event load. Finally, in our case the same event load is applied consistently during the whole experiment, while such values on an actual trace are only temporary and would typically be surrounded by smaller values.

Consequently, this experiment shows that HINT is able to deal with a number of events per second higher than the maximum value observed in real traces, which validates its scalability in terms of number of events per second.

8.1.2 Realism

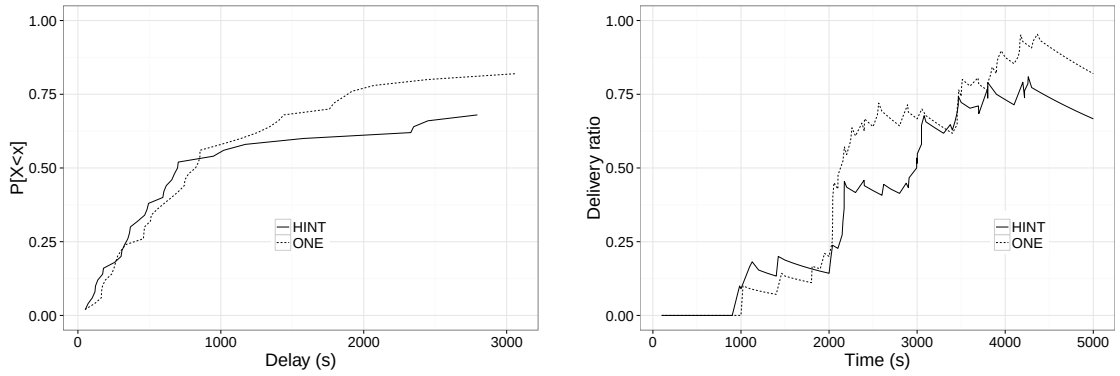
Evaluating the realism will be done by comparing the performance results obtained with HINT to the results obtained with the ONE [47] v. 1.6.0, using the same scenario parameters and (inter)contact inputs. HINT already offers the delay CCDF, hop count, drop number and delivery ratio through the Monitoring interface. Here, we will take advantage of the ability of HINT to read real traces by using the 5000 first seconds of the Rollernet dataset, considering only the 62 internal nodes. 0-second contacts are extended to 1 second, and no extreme values are removed. We choose this dataset because this is the shortest one among the four considered in this thesis, which makes repeating experiments easier.

Because routing, buffer management and timestamps are managed by the Core Emulator, this experiment will again be conducted with a network comprising only virtual nodes. To be able to perform the comparison, we need to choose parameters which are supported both by HINT and the ONE. TTL will be set to the simulation duration in ONE, and disabled in HINT. Connection bandwidth is set to 250 kB (the default in ONE), message size to 10 kB. We choose the simple forward routing protocol (called FirstContact in ONE), buffer size is set to 1 MB and the management policy is drop_oldest in its Avoid source packets version, because this is the policy implemented in the ONE[1]. Finally, we will consider communication between nodes 27 and 38, as in Section 5.5, with 1 message generated from node 27 to node 38 every 100 seconds.

The results are presented in Figure 8.2 for the delay distribution and delivery ratio over time. Indeed, with the hypotheses taken for this experiment, the routing protocol

[1] Avoid Source Packets is an interpretation here, the source code precisely reads exclude messages being sent . At the time of this writing, additional buffer management policies for the ONE are in a pull request on GitHub.

selected virtually removes drops, due to the high buffer size. We see that for this experiment, HINT gives a bigger delay and a smaller delivery ratio than the ONE. For example, at the end of the experiment, ONE finds 41 delivered messages vs 34 for HINT. One explanation for this can be found in the difference of implementations between HINT and ONE. In HINT, in order to limit queue polling, we chose to process messages only at the beginning of contacts, whereas the ONE does this at each time step (1 second in the present experiment) for all nodes. Consequently, when a message is created on a node and with the routing protocol chosen, ONE will transfer it immediately if a connection is available, while HINT will wait for the next contact start. Furthermore, in ONE the nodes are updated in sequence with increasing IDs, while HINT does this in the order of events, for which the scheduler used can create some randomness with events all scheduled at the same second. In conclusion, we can consider the results provided by HINT to be more pessimistic than those obtained with the ONE.



(a) Delay distribution

(b) Delivery ratio

Figure 8.2: Comparison of network performance metrics obtained with HINT and with the ONE.

For completeness, we have to mention that this example only illustrates one specific experiment, so the results presented here may not necessarily apply to all situations.

8.1.3 Considerations on real nodes

For the moment, HINT was only run with virtual nodes. Because one of the main features of an emulator is its ability to also accommodate real nodes, we now discuss the integration of real nodes.

During our experiments with real nodes, we noticed that the connection delay between the emulator and the real devices is negligible compared to the event durations managed during the experiments. Then, we should note that the design of HINT leaves only the User Link Layer on the real nodes, the rest of the message processing being done in the Core. As a result, the fact that a given node is real does not impact on this processing. We should also mention that the User Link Layer performs a direct connec-

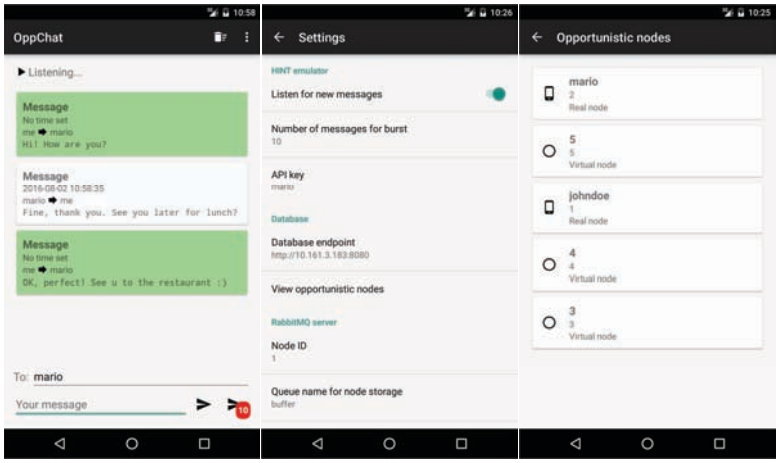


Figure 8.3: Views of the OppChat application: messages, settings and node list.

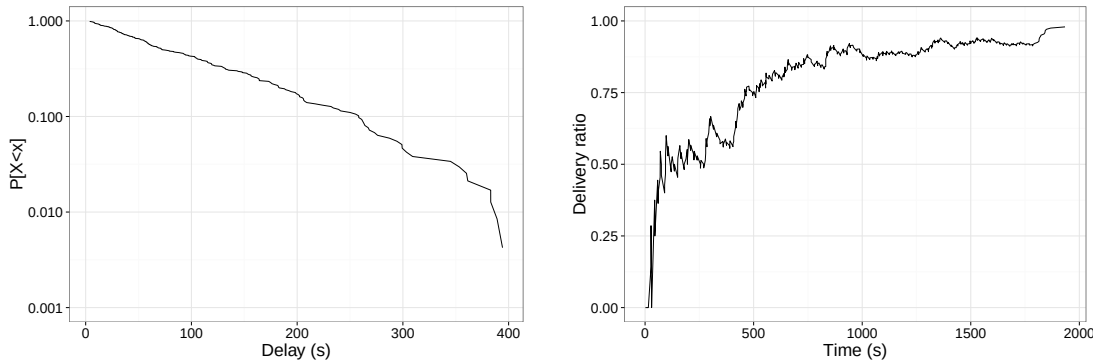
tion with the Message Broker, without further operation on the messages. Finally, the centralized approach chosen for HINT also means that real nodes do not create any issue with additional delays for monitoring data, or with clock synchronization as discussed in Section 7.5.

8.2 Opportunistic application development

In Section 8.1, we showed that the emulator could successfully handle event loads and node numbers of an order of magnitude compatible with the study of DTNs, and assessed the realism of the performance results provided by HINT. In this section, we use the now validated HINT emulation system and exploit it to test the custom-made opportunistic mobile application described below.

As a first candidate for HINT, we design OppChat, an opportunistic chat application. It can either broadcast a message to all nodes in the network, by creating one copy for each node, or deliver a message to a single specific node. Messages are currently restricted to character strings, bigger content like pictures are left for future work or other applications. Note that internally, the Core Emulator, Message Broker and Database also assume message payloads as strings. Consequently, a conversion such as Base64 would be required to exchange binary payloads in HINT. To help us achieve higher network loads without having to type loads of messages (pun intended), we also provide in OppChat an extra button to send a burst of 10 identical messages at once. Several screenshots of the application are provided in Figure 8.3. As detailed in Chapter 7, this application is integrated in the HINT emulation system through the use of the methods provided by the User Link Layer, along with the required settings.

In this section, we present two experiments involving OppChat: a discussion between two users through an opportunistic network, and a situation where all nodes send regular status updates to a single central node. For simplicity, and unless mentioned otherwise,



(a) Delay CCDF

(b) Delivery ratio

Figure 8.4: Results for the delay and delivery ratio for the Discussion test case.

no TTL will be applied, and we will consider all nodes equal regarding their bandwidth (250 kB), buffer size (1 MB) and management policy (drop_oldest, avoid source packets). Routing protocol will be set to forwarding. CT and ICT parameters will also be homogeneous.

8.2.1 Discussion between users

In this first test case, 2 real nodes will discuss with each other through an emulated opportunistic network of 48 virtual nodes, for a total of 50 nodes. This discussion translates into a traffic of 1 message of 1 kB every 15 seconds in each direction, with a total experiment duration of 30 minutes. This load is voluntarily made high: in practice, real users are highly unlikely to manage to type 1 kB of text every 15 seconds. For simplicity, no other traffic will be generated in the network, although this possibility is supported by HINT. Note that this traffic model is very close to the assumption usually taken for analytical models of a single node pair exchanging messages in an otherwise empty network, as discussed in Chapter 2. We use homogeneous exponentially distributed CT and ICT with parameters $\lambda = 0\ 5$ and $0\ 01$ respectively, generated with Henget. We choose the exponential distribution not only because it is widespread in DTN literature, but also due to the simple relationship between the mean value and the λ parameter. To make the situation slightly worse, we also remove every direct contact between the two nodes representing the real devices, respectively numbered as 1 and 2 in the experiment.

The results are presented in Figure 8.4. Note that unlike the previous experiment, we use the delay CCDF. No drops are registered: this result was expected, given the size of the buffers used. We notice that for this discussion, and although messages are being created during the whole experiment duration, the final delivery ratio obtained through HINT is of approximately 98%. For the delay, we obtain a median value of 72.1 s with the minimum and maximum recorded values being 4 s and 572 s respectively.

Considering the shape of the distribution, shorter values are more likely. On a more practical note, it would be impossible, considering the delays obtained, to effectively make an actual conversation by exchanging messages every 15 seconds. Regarding the delivery ratio, the small oscillations which can be noticed on the curve are caused by the creation of new messages during the experiment.

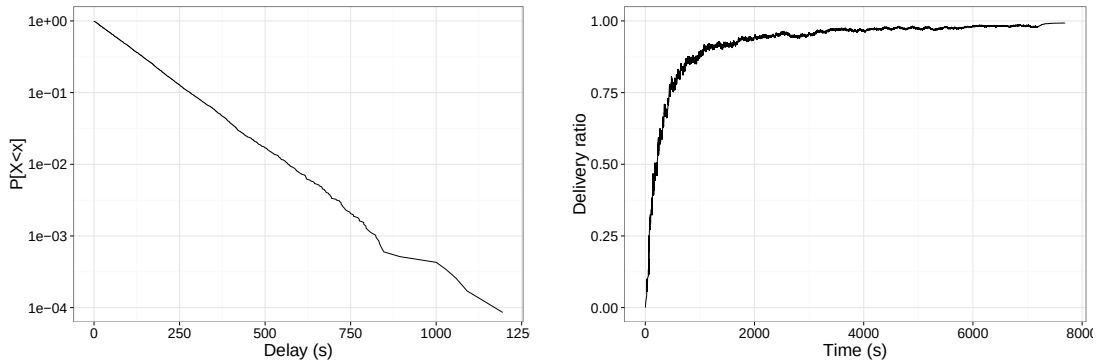
8.2.2 Report to HQ

For this second test case, we consider a scenario with an opportunistic network of 50 nodes split into 49 virtual nodes and 1 real Android device. Pairwise intercontact events are generated with Henget for all pairs according to an homogeneous exponential distribution with $\lambda = \frac{1}{120}$, while we set contact durations to a constant value of 2 seconds. Again, all nodes route their messages using the forward protocol (single copy).

Unlike the previous subsection, this time all nodes will periodically send messages to a single real node every 30 seconds. Such a situation may arise for example in a sensor network with only one collection station, or on a battlefield with soldiers having to send status updates to the headquarters. The aim here is to get an overview of the current situation from the headquarters which is as precise as possible. Because we are using DTNs, the exact spatial repartition of nodes will affect spatial precision of the measurement, while the inherent delays and packet losses will affect temporal precision. In this work, we will only be interested in the temporal precision, as represented by the observed delay distribution and delivery ratio of messages. Indeed, the contact-based emulation provided by HINT does not allow us to consider the location of nodes directly, but only its impact on the recorded contacts and intercontacts.

We set the experiment duration to 2 hours. The message size is 10 kB, which should be enough for each node to submit a detailed report of its situation (position, health and equipment status...). Apart from these status messages, no other traffic happens in the network. With 49 pairs exchanging messages and a 2-hour duration, this experiment represents a much higher load for HINT than the ones presented earlier.

The results can be found in Figure 8.5. We find that on the linear-log scale used to plot the delay CCDF, the curve is almost a straight line, with values ranging from 1 to 1262 seconds and a median of 86 seconds. In this case, setting a TTL of 15 minutes would mean losing 0 04 % of the messages. It is also interesting to see that the delivery ratio increases very quickly for the 1000 first seconds of the simulation before reaching an almost steady state. At the end of the experiment, 99 % of the messages have been delivered. According to these results, which were obtained using a single-copy routing protocol (forwarding), it may not be useful to switch to a multi-copy routing to obtain better results. Finally, despite representing a much higher load than previous experiments, we did not notice any issue with event delays, database accesses or queues while performing this experiment. However, there was a noticeable delay to generate and display the delay and delivery ratio data in the monitoring system.



(a) Delay

(b) Delivery ratio

Figure 8.5: Results in terms of delay and delivery ratio for the Report to HQ use case.

8.3 Discussion

We already discussed some considerations related to the implementation of HINT in Section 7.5. In this section, we will focus instead on possible future experiments and platform enhancements.

8.3.1 Other possible experiments

Although this chapter presented several experiments aiming at evaluating our emulator, there are in fact many other assessments which can be done.

First, we can check the actual transparency offered by HINT in terms of impact on the derived performance results. To do so, we would have to compare the predicted performance with the real one, which means setting up a field trial. Despite being interesting, this experiment is unlikely to be made in the near future, considering the issues of repeatability and cost involved by field trials. These issues were already mentioned at the end of Chapter 2.

Then, we can try to evaluate the scalability of HINT regarding node numbers in addition to events. Note that the amount of messages which can be in the network at a given time is not limited by the amount of RAM available on the computer running the Core Emulator, because RabbitMQ performs paging on the hard disk as necessary. However, we had to increase the file descriptor limit for RabbitMQ to handle a sufficient number of queues in the most demanding scenarios. Concerning the database part, we use MongoDB, which should be able to handle at least 1000 nodes without any issue. Trying to design an experiment for this particular test case proves especially tricky. Although one can choose to study a worst case scenario by selecting ooding routing and infinite buffers, other factors like contact opportunities and message frequencies can also in uence the quantity of messages generated, which in turn in uences the memory footprint of the system.

We chose to verify realism using a comparison with the ONE [47] simulator, but

other comparisons are also possible. Following the example of the TUNIE testbed [50], we could have used an analytical model like the one introduced in Chapter 4 for the drop ratio. Unfortunately, the comparison may not be as meaningful as desired, considering that this model only provides the average value. Another candidate here can be the model from [27] for the delay using Binary Spray and Wait routing, which requires the implementation and testing of this specific routing algorithm into HINT. Regardless of the model chosen, we would have to face the limitations detailed in Chapter 2, especially the number of metrics provided. Instead of models, we could also consider here a comparison with the results of a real experiment, as already expressed above for checking transparency.

Finally, it will be interesting to use other applications instead of the chat software employed here, especially if the message size produced greatly differ. A picture exchange application could be an example to produce much bigger message sizes than short, text-based status updates.

8.3.2 Future developments

A future development of the system could be to allow a varying number of nodes during the experiment. Indeed, this was the initial motivation behind the use of a database in HINT. However, a varying number poses several issues. The first one is ID allocation for new nodes, because colliding IDs may lead to undesired behaviors and possibly wrong results. Creation of node entries in the database and queues in the message broker also need to be done dynamically during the experiment, while it is currently done only once at startup. Finally, the biggest issue comes from the reproducibility of node number changes, for which precise timing may not be easily achievable (especially if such changes result from uncontrolled human mobility of users entering/leaving the experimental network).

Another possible enhancement could be to add more details to connection impairments, as KauNet [62] and KauNetEm [31] can do. In the current implementation of HINT, the only effect supported in addition to disconnections is a constant bandwidth limitation, which can be modified during the experiment if needed with the web-based tuning interface. Although potentially valuable, adding more complex impairments would require substantial changes to our Core Emulator. Designing and preparing the input patterns describing these changes would also be necessary, which adds even more inputs and complexities to the system.

8.4 Conclusion

In this chapter, we went beyond the implementation of HINT and used it for various experiments. First, the validation part showed that HINT can successfully deal with a number of events per second higher than what is observed in real datasets. Then, we evaluated the realism of HINT by comparing the performance results obtained with those from a simulator in terms of delay and delivery ratio. We noticed that the results pro-

vided by HINT are more pessimistic than with the ONE, due to different implementation choices.

In a second time, we designed a DTN chat application named OppChat, and integrated it in HINT using the User Link Layer. This allows us to assess the performance of the actual application over a DTN, and if needed to tune some parameters (like the buffer size or buffer management policy) to achieve optimal results without resorting to a real-world deployment. In this chapter, we proposed two usage scenarios for OppChat. The study of a discussion between two users through an opportunistic network showed a very good delivery ratio, but the delays experienced would not allow to conduct an actual discussion with the message generation parameters considered. Then, we considered another case where all nodes report their status to a single node. For this second case, we notice that with the intercontact distribution considered, it is possible to achieve a delivery ratio close to 1 even with a single-copy routing protocol.

Future work will include using the emulator with different applications and releasing HINT to the public as open-source software, so it can benefit to the community and receive feedback or further developments.

Chapter 9

Conclusion and future directions

Contents

DTNs and opportunistic networks are a promising alternative to infrastructure-based networks. Even in the presence of ubiquitous connectivity and comprehensive network coverage, opportunistic networks are still interesting in specific contexts like censorship evasion, data o oading or natural disasters. Yet, there is still no easy way to predict the performance of such networks, or of applications running on top of them. Furthermore, none of the existing proposals specifically target application development. Based on the requirements for development, namely the ability to execute real applications and the need for a lightweight tool, we turn to emulation.

The first part of the thesis dealt with the production of inputs for an emulation system. We first proposed an analytical model for the drop ratio which is an underrepresented metric in the literature. However, this model only allows to derive one metric, and requires many simplifying hypotheses which may not hold in reality. This clearly limits the scope of use for such a model. Then, we turned to real contact traces. We studied the in uence of the preliminary hypotheses on the result of the statistical analyses often conducted to scale traces, and noticed that some of them like the removal of the shortest contacts have a strong in uence. An extension to the in uence on performance results was also presented, which showed less conclusive findings. We then went deeper in this study on assumptions, by covering the whole work ow, from collection to scaling. After listing the assumptions and filters which are applied at the different stages, we provided recommendations for three use cases (social analysis, performance assessment, statistical analysis). We also showed that due to the dependencies between assumptions,

the possibility of using synthetic models should not be neglected.

This study of possible inputs allowed us in a second part to propose an emulator aimed at helping developers build applications for opportunistic networks. Indeed, we found out that none of the existing approaches really allows to include real applications in the loop, or are adapted to development environments. First, we listed requirements for our system, then derived an architecture and the corresponding implementation in Python. The result is HINT (Hint Is Not a Testbed), a hybrid lightweight emulation system which combines real and simulated nodes in a common network. We validated HINT by evaluating its ability to scale in terms of number of events per second, and compared the performance results with data from a simulator. This comparison showed that our results were more pessimistic. Finally, the validation of HINT allowed us to perform experiments using a custom-made Android opportunistic chat application. Two use cases were considered, namely a discussion between two users and a situation where all nodes have to report to a single node. In both cases, the delivery ratio was found to be very close to 1. We also noticed in the discussion case that the delays experienced would require a reduction of the message frequency in order to be able to conduct an actual discussion.

At this point, we unfortunately have to mention some recent events. 2016 saw the closing of Delay-Tolerant Network Working Group (DTNWG) and Delay-Tolerant Network Research Group (DTNRG), which were the work groups linked with DTN standardization. Even in the context of natural disasters, where delay-tolerant networking could be an especially good fit, temporary infrastructure-based networks may in fact be used instead. An example is the CaribeWave tsunami simulation in the Caribbean [36], where the main focus is to restart a limited Wifi and cellular network infrastructure and not setting up DTN functionality. Consequently, we currently believe that although DTNs and opportunistic networking are interesting fields, their applicability is likely to remain restricted for the moment to a few specific use cases like space communications or censorship evasion.

9.1 Future work

During the course of this thesis, we tried to explore various paths, some of which could be extended further. Hence, this section lists areas that we think could be worth investigating for future contributions.

9.1.1 Emulation function

At the very beginning of the Ph.D. work, our goal was to be able to represent the whole network as a mathematical function, with inputs (CTs, ICTs, node number, total time...) and outputs (hopefully, performance results such as delay or delivery ratio). Then, these outputs would be fed to a link emulator in order to emulate the network. This closely resembles the analytical performance models such as the one presented in Chapter 4. However, every existing performance model derived so far requires a

tremendous amount of assumptions to become mathematically tractable, and also a lot of time to perform the initial derivations. Hence, we did not investigate a lot this research direction. If such a functional representation of the network is feasible, it is highly likely to achieve a rather low realism compared to what an emulation system can provide. We also noticed that several models need to be used, up to one per desired network metric, thus quickly making this approach unpractical. On the contrary, the speed of use (once the model derivation is made) would be of great interest, as with all the theoretical models. Being able to prove that this *emulation function* is not possible in the general case would represent a valuable contribution.

9.1.2 Improvements to HINT

We proposed in this work the HINT DTN emulation system, specifically targeted to developers of opportunistic applications. However, several enhancements to the platform are still feasible. Indeed, the modular architecture we propose could allow us to go from the lightweight, development-oriented tool we promote to a full testbed. While this approach would be less suitable for developers, it would increase realism and the total number of nodes we could manage in our system. It will also allow to use a less abstract node model: indeed, nodes are currently only represented as a message buffer. One possibility would be to use virtual machines instead, as in [71]. As with all currently existing testbeds, some time would then have to be invested in the development of a proper management system, to handle both metric collection, node activation and application setup. An example of such a system is OMF [68].

Another improvement possibility is to offer more choice in the parameters, such as more routing protocols or other buffer management techniques. One good source of inspiration for additional algorithms to implement is the Adyton simulator [60], which offers an unprecedented choice of routing protocols, congestion control mechanisms and buffer management techniques. It unfortunately has the drawback of being only able to work with preprocessed real traces at the time of this writing, excluding external nodes, and does not include any mobility model.

A third area of evolution is to add more exibility to the platform, by supporting a varying number of nodes. The design decision for a user database was initially driven by this requirement. In such a configuration, real nodes can seamlessly join or leave during the experiment. This could be for example caused by real motion of the users carrying the real nodes on the campus. However, it should be noted that this feature would suffer from a huge issue of reproducibility, because it would be quite hard to exactly reproduce connections of nodes with the emulator, let alone the actual mobility behind these connections. This exibility is also going to be really challenging for several aspects of our emulation system, including routing management or monitoring.

A fourth improvement could be to include existing third-party software components. In order to perform a more precise link emulation, including bandwidth changes, reordering or bit errors, we could use the KauNetEm [31] component developed at Karlstad University. However, such a level of detail is not yet required. Also note that this component will also require dedicated emulation condition inputs, as mentioned in

Chapter 2. At the node level, it may also be desirable to replace the User Link Layer presented in Chapter 7 by one of the existing DTN stacks like ION [11], DTN2 [25] or IBR-DTN [75]. Although not currently deployed on any user platform, these stacks have the major advantage of implementing the existing RFCs defined for delay-tolerant networking. We would choose IBR-DTN, because among the three existing DTN stacks, this is the only one which currently supports Android. It also offers 3 open source applications running on top of it and ready to use: a chat system (Whisper), a voice message service (Talkie) and a file transfer system (Sharebox). New applications can be created if necessary by using the API provided by the authors.

Finally, we plan to release our emulator as an open source tool, in accordance with the requirement on Availability (E7) from Section 7.2. This will facilitate its adoption and further development by the community.

9.1.3 New scaling approaches

In Section 6.6, we provided some recommendations on what an ideal scaling approach would have to take into account. Especially, considering pairwise metrics and validating the approach with network performance results are the main requirements we advocate. Proposing this tool, which we nicknamed EMO++ during the course of the present work, would prove to be a valuable contribution. Indeed, it would remove one major drawback of the use of contact traces, namely their limited range in terms of node number and recorded time. Even if the development of the tool ultimately fails, it will likely uncover additional requirements for the system, which is also a useful result.

Another idea could involve splitting the trace into several parts, and trying to model each one separately [42]. On a typical work day, this would mean for example using a different model for commuting in the morning and for the lunch. Our observations made on the Infocom 2005 trace [76] showed that the time slices corresponding to breaks and lunches recorded the highest numbers of contacts, as opposed to paper sessions and keynotes where people are less mobile. However, this approach suffers from a big issue: restricting the time scope available for analysis always means reducing the amount of data available for it. This includes contacts, intercontacts and visible node pairs. Furthermore, all data (especially intercontacts) which is longer than the time scope considered would disappear, although we noticed in Section 6.5 that these longer values played a major role. Considering the study conducted in Chapter 6, the risk is that by trying to perform a more precise analysis by slicing time, one may end up with a less precise analysis due to the smaller amount of data used.

Finally, we can observe that all the approaches of the literature [83, 16], as well as our study in Chapter 6, deal with CT and ICT separately, as if there was no relationship between these two values. Yet, intuitively, the sum of CT and ICT for a given pair cannot exceed the length of the contact trace. Another trivial requirement is that a contact or intercontact cannot have a negative duration, which restricts the range of distributions that can be used for modeling. In the synthetic data generation phase of the EMO approach [83], the authors implement a time limit to ensure that the total values of event lengths cannot go past the trace length. We saw in Section 6.5 that

using an event number limit instead could easily lead to exceeding the trace length. The ability to consider CT and ICT together, and the approach to use to achieve this goal are an open question.

9.1.4 Representativity of statistical analysis

The first part of this thesis dealt with trace use as a whole, with a particular focus on real trace filtering and statistical analysis. When interested in network performance, we highlighted two possible work ows, represented in Figure 9.1. In both cases, we start with the raw [1] material, which should be followed by a filtering step as discussed in detail in Chapter 6. Then, two processes are possible:

play filtered traces directly in a simulator to get network performance metrics, or

feed these filtered traces to a statistical analysis process like EMO [83], then generate a synthetic trace based on this statistical characterization. Use this synthetic trace in a simulator to obtain the performance metrics as before. We assume here that the synthetic trace, unlike the real one, will not need a filtering phase before being fed to the simulator.

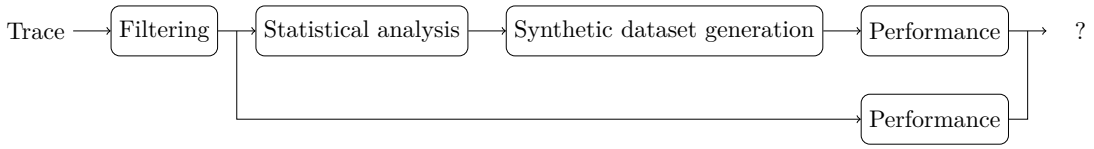


Figure 9.1: Explanation of the 2 possible work ows when working with contact traces.

If we go back, filtering was covered in Chapters 5 and 6, while statistical analysis and the lower path of the graph were studied in Chapter 5. Also note that apart from the Filtering step, the remainder of the upper path roughly matches the work ow from EMO [83]. Due to a lack of time, and considering the limitations expressed in Section 6.5 regarding the process, we did not completely explore the upper path after the statistical analysis step. Consequently, we did not manage to perform the comparisons of the results provided by both approaches, suggested by the at the right of the diagram. This would however represent a highly valuable contribution: indeed, failure of the statistical analysis process would be a serious issue for all existing trace scaling proposals, and prevent the development of the scaling approach presented in the previous subsection.

9.1.5 Cross-disciplinary approach

The whole thesis work has been conducted with network characterization and application development in mind. However, DTNs and opportunistic networks are not only of interest for network scientists. Because devices are typically worn by users, which

[1] See Chapter 6 for a justi cation of the quote marks.

exhibit social behaviors, this field of study is also of interest for researchers interested in social sciences. The link with real traces and social analysis is consequently very strong [41]. As mentioned in Chapter 6, it is still unclear to us whether an approach which can accurately reproduce network performance would also accurately reproduce social characteristics. We believe that this issue would be worth investigating.

To cap it all, the work presented in this thesis introduces a new tool for DTN performance evaluation, the HINT emulation system, which can handle real and simulated nodes at the same time. HINT aims to ease the development of applications for opportunistic networks. This work also raised many issues regarding input generation from traces, especially considering the numerous hypotheses and assumptions associated with this process.

List of publications

The following publications have been produced during the course of the thesis work.

G. Baudic, T. Perennou and E. Lochin. Revisiting Pitfalls of DTN Datasets Statistical Analysis. In *ACM CHANTS 2014*, pages 73-76, September 2014.

V. Ramiro, D.-K. Dang, G. Baudic, T. Perennou and E. Lochin. A Markov chain model for drop ratio on one-packet buffers DTNs. In *IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC)*, pages 1-6, June 2015.

G. Baudic, T. Perennou and E. Lochin. Following the Right Path: Using Traces for the Study of DTNs. In *Elsevier Computer Communications*, vol. 88C, pages 25-33, August 2016.

G. Baudic, A. Auger, V. Ramiro and E. Lochin. HINT: from Network Characterization to Opportunistic Applications. In *ACM CHANTS 2016*, pages 13-18, October 2016.

A. Auger, G. Baudic, V. Ramiro and E. Lochin. Demo: Using the HINT Network Emulator to Develop Opportunistic Applications. In *ACM CHANTS 2016*, pages 35-36, October 2016.

Bibliography

[1] M. Abdulla and R. Simon. The impact of the mobility model on delay tolerant networking performance analysis. In *Simulation Symposium, 2007. ANSS 07. 40th Annual*, pages 177 184, March 2007.

[2] Ahmad Al Hanbali, Arzad A. Kherani, and Philippe Nain. Simple models for the performance evaluation of a class of two-hop relay protocols. In *Proceedings of the 6th International IFIP-TC6 Conference on Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, NETWORKING'07, pages 191 202. Springer-Verlag, 2007.

[3] Nils Aschenbruck, Raphael Ernst, Elmar Gerhards-Padilla, and Matthias Schwamborn. Bonnmotion: a mobility scenario generation and analysis tool. In *3rd International ICST Conference on Simulation Tools and Techniques*. ICST, 5 2010.

[4] Nilanjan Banerjee, Mark D. Corner, Don Towsley, and Brian N. Levine. Relays, Base Stations, and Meshes: Enhancing Mobile Networks with Infrastructure. In *ACM MobiCom 08*, pages 81 91, 2008.

[5] Farid Benbadis and Jeremie Leguay. CRAWDAD dataset upmc/rollernet (v. 2009-02-02). Downloaded from http://crawdad.org/upmc/rollernet/20090202, February 2009.

[6] R. Beuran, S. Miwa, and Y. Shinoda. Network emulation testbed for DTN applications and protocols. In *INFOCOM, 2013 Proceedings IEEE*, pages 3441 3446, April 2013.

[7] Razvan Beuran. *Introduction to network emulation*. Pan Stanford Publishing, 2013.

[8] Razvan Beuran, Shinsuke Miwa, and Yoichi Shinoda. Making the Best of Two Worlds: A Framework for Hybrid Experiments. In *ACM WiNTECH 12*, 2012.

[9] D. Bittencourt, E. Mota, E. Nascimento Silva, and C.B. Souza. Towards realism in DTN performance evaluation using virtualization. In *Wireless Days (WD), 2013 IFIP*, pages 1 7, November 2013.

[10] Chiara Boldrini, Marco Conti, and Andrea Passarella. Modelling social-aware forwarding in opportunistic networks. In *Performance Evaluation of Computer and Communication Systems. Milestones and Future Challenges*, volume 6821 of *Lecture Notes in Computer Science*, pages 141 152. Springer Berlin Heidelberg, 2011.

[11] S. Burleigh. Interplanetary overlay network: An implementation of the DTN bundle protocol. In *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE*, pages 222 226, Jan 2007.

[12] Jose M. Cabero, Virginia Molina, Inigo Urteaga, Fidel Liberal, and Jose L. Martin. CRAWDAD dataset tecnalia/humanet (v. 2012-06-12). Downloaded from http://crawdad.org/tecnalia/humanet/20120612, June 2012.

[13] José María Cabero, Virginia Molina, Iñigo Urteaga, Fidel Liberal, and José Luis Martin. Acquisition of human traces with Bluetooth technology: Challenges and proposals. *Ad Hoc Networks*, 12:2 16, January 2014.

[14] José María Cabero, Iñigo Urteaga, Virginia Molina, Fidel Liberal, and José Luis Martin. Reliability of Bluetooth-based connectivity traces for the characterization of human interaction. *Ad Hoc Networks*, 24, Part A:135 146, January 2015.

[15] Han Cai and Do Young Eun. Crossing Over the Bounded Domain: From Exponential to Power-Law Intermeeting Time in Mobile Ad Hoc Networks. *IEEE/ACM Transactions on Networking*, 17(5):1578 1591, October 2009.

[16] Roberta Calegari, Mirco Musolesi, Franco Raimondi, and Cecilia Mascolo. CTG: A connectivity trace generator for testing the performance of opportunistic mobile systems. In *ACM ESEC/FSE07*, Dubrovnik, Croatia, September 2007.

[17] Tracy Camp, Jeff Boleng, and Vanessa Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483 502, 2002.

[18] Marta Carbone and Luigi Rizzo. Dummynet revisited. *SIGCOMM Comput. Commun. Rev.*, 40(2):12 20, April 2010.

[19] A. Chaintreau, Pan Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *Mobile Computing, IEEE Transactions on*, 6(6):606 620, June 2007.

[20] Augustin Chaintreau, Abderrahmen Mtibaa, Laurent Massoulie, and Christophe Diot. The diameter of opportunistic mobile networks. In *Proceedings of the 2007 ACM CoNEXT Conference*, CoNEXT '07, pages 12:1 12:12, 2007.

[21] Xinjie Chang. Network simulations with OPNET. In *Proceedings of the 31st conference on Winter simulation: Simulation a bridge to the future-Volume 1*, WSC '99, pages 307 314. ACM, 1999.

[22] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661 703, 2009.

[23] Vania Conan, Jérémie Leguay, and Timur Friedman. Characterizing pairwise intercontact patterns in delay tolerant networks. In *Autonomics 07*, pages 19:1 19:9, 2007.

[24] Emmanuel Conchon, Tanguy Pérennou, Johan Garcia, and Michel Diaz. W-NINE: A Two-Stage Emulation Platform for Mobile and Wireless Systems. *EURASIP Journal on Wireless Communications and Networking*, 2010(1), December 2009.

[25] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho, and R. Patra. Implementing delay tolerant networking. Technical Report IRB-TR-04-020, Intel Research, Dec 2004.

[26] Michael Demmer, Joerg Ott, and Simon Perreault. Delay-Tolerant Networking TCP Convergence-Layer Protocol. IRTF RFC 7242, June 2014.

[27] R. Diana and E. Lochin. Modelling the delay distribution of binary spray and wait routing protocol. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1 6, June 2012.

[28] Nathan Eagle and Alex (Sandy) Pentland. CRAWDAD dataset mit/reality (v. 2005-07-01). Downloaded from http://crawdad.org/mit/reality/20050701, July 2005.

[29] M. P. Freeman, N. W. Watkins, E. Yoneki, and J. Crowcroft. Rhythm and randomness in human contact. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 184 191, Aug 2010.

[30] Wei Gao, Qinghua Li, Bo Zhao, and Guohong Cao. Multicasting in delay tolerant networks: A social network perspective. In *Proceedings of the Tenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '09, pages 299 308, 2009.

[31] Johan Garcia and Per Hurtig. Kaunetem: Deterministic network emulation in linux. In *Proceedings of netdev 1.1*, February 2016.

[32] Eugenio Giordano, Lara Codecà, Brian Geffon, Giulio Grassi, Giovanni Pau, and Mario Gerla. MoViT: The Mobile Network Virtualized Testbed. In *ACM VANET 12*, 2012.

[33] Samo Grasic and Anders Lindgren. An analysis of evaluation practices for DTN routing protocols. In *Proceedings of the Seventh ACM International Workshop on Challenged Networks*, CHANTS '12, pages 57 64, 2012.

[34] Robin Groenevelt, Philippe Nain, and Ger Koole. The message delay in mobile ad hoc networks. *Performance Evaluation*, 62(1-4):210 228, October 2005.

[35] R. Gunasekaran, V. Mahendran, and C. Siva Ram Murthy. Performance modeling of delay tolerant network routing via queueing petri nets. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1 6, June 2012.

[36] Camille Gévaudan. CaribeWave : les geeks et le tsunami - Libération (in french). http://www.liberation.fr/futurs/2016/03/19/caribewave-les-geeks-et-le-tsunami_1440465, March 2016.

[37] Z.J. Haas and T. Small. A new networking model for biological applications of ad hoc sensor networks. *Networking, IEEE/ACM Transactions on*, 14(1):27 40, Feb 2006.

[38] A.A. Hanbali, P. Nain, and E. Altman. Performance of ad hoc networks with two-hop relay routing and limited packet lifetime. In *Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on*, pages 295 296, June 2006.

[39] Thomas R. Henderson, Mathieu Lacage, George F. Riley, C. Dowell, and J. Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 14, 2008.

[40] A. Hess, E. Hyytia, and J. Ott. Efficient neighbor discovery in mobile opportunistic networking using mobility awareness. In *Communication Systems and Networks (COMSNETS), 2014 Sixth International Conference on*, pages 1 8, Jan 2014.

[41] T. Hossmann, Thrasyvoulos Spyropoulos, and F. Legendre. A complex network analysis of human mobility. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 876 881, April 2011.

[42] Wei-jen Hsu, Thrasyvoulos Spyropoulos, K. Psounis, and A. Helmy. Modeling Spatial and Temporal Dependencies of User Mobility in Wireless Mobile Networks. *IEEE/ACM Transactions on Networking*, 17(5):1564 1577, October 2009.

[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 WDTN 05*, pages 244 251, 2005.

[44] Yin-Ki Ip, Wing-Cheong Lau, and On-Ching Yue. Performance modeling of epidemic routing with heterogeneous node types. In *Communications, 2008. ICC 08. IEEE International Conference on*, pages 219 224, May 2008.

[45] Samuel Jero, Shawn Ostermann, and Hans Kruse. Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP). IRTF RFC 7122, October 2015.

[46] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnovi . Power law and exponential decay of intercontact times between mobile devices. *Mobile Computing, IEEE Transactions on*, 9(10):1377 1390, October 2010.

[47] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *ICST Simutools 09*, pages 55:1 55:10. ICST, 2009.

[48] M. Kim, D. Kotz, and S. Kim. Extracting a Mobility Model from Real User Traces. In *IEEE INFOCOM 2006*, pages 1 13, April 2006.

[49] David Kotz, Tristan Henderson, Ilya Abyzov, and Jihwang Yeo. CRAWDAD dataset dartmouth/campus (v. 2009-09-09). Downloaded from http://crawdad.org/dartmouth/campus/20090909, September 2009.

[50] Y. Li, P. Hui, D. Jin, and S. Chen. Delay-tolerant network protocol testing and evaluation. *Communications Magazine, IEEE*, 53(1):258 266, January 2015.

[51] Anders Lindgren and Pan Hui. The Quest for a Killer App for Opportunistic and Delay Tolerant Networks: (Invited Paper). In *Proceedings of the 4th ACM Workshop on Challenged Networks*, pages 59 66, New York, NY, USA, 2009. ACM.

[52] J. Liu, S. Mann, N. Van Vorst, and K. Hellman. An Open and Scalable Emulation Infrastructure for Large-Scale Real-Time Network Simulations. In *IEEE INFOCOM 2007*, May 2007.

[53] Emmanuel Lochin, Tanguy Pérennou, and Laurent Dairaine. When should I use network emulation? *annals of telecommunications - annales des télécommunications*, 67(5-6):247 255, July 2011.

[54] K. Maeda, K. Nakata, T. Umedu, H. Yamaguchi, K. Yasumoto, and T. Higashinoz. Hybrid Testbed Enabling Run-Time Operations for Wireless Applications. In *PADS 08*, June 2008.

[55] V.K.C. Manam, V. Mahendran, and C. Siva Ram Murthy. Performance modeling of routing in delay-tolerant networks with node heterogeneity. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1 10, Jan 2012.

[56] Alireza K. Monfared, Mostafa H. Ammar, and Ellen W. Zegura. Plausible Mobility Inference from Wireless Contacts Using Optimization. In *ACM CHANTS 13*, pages 7 12, 2013.

[57] Johannes Morgenroth, Sebastian Schildt, and Lars Wolf. HYDRA: Virtualized distributed testbed for DTN simulations. In *Proceedings of the Fifth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, WiNTECH '10, pages 71 78, 2010.

[58] Mirco Musolesi and Cecilia Mascolo. Mobility Models for Systems Evaluation - A Survey. In *Middleware for Network Eccentric and Mobile Applications*. Springer-Verlag, 2008.

[59] A. Nayebi and G. Karlsson. Beaconing in Wireless Mobile Networks. In *IEEE WCNC 2009*, pages 1 6, April 2009.

[60] Nikolaos Papanikos, Dimitrios-Georgios Akestoridis, and Evangelos Papapetrou. CRAWDAD toolset tools/simulate/uoi/adyton (v. 2016-04-21). Downloaded from http://crawdad.org/tools/simulate/uoi/adyton/20160421, April 2016.

[61] A. Passarella and M. Conti. Analysis of individual pair and aggregate intercontact times in heterogeneous opportunistic networks. *IEEE Transactions on Mobile Computing*, 12(12):2483 2495, Dec 2013.

[62] Tanguy Pérennou, Anna Brunstrom, Tomas Hall, Johan Garcia, and Per Hurtig. Emulating opportunistic networks with kaunet triggers. *EURASIP Journal on Wireless Communications and Networking (EURASIP JWCN)*, 2011:1 15, 2011.

[63] Tiphaine Phe-Neau, Marcelo Dias de Amorim, and Vania Conan. Fine-grained intercontact characterization in disruption-tolerant networks. In *IEEE Symposium on Computers and Communications*, pages 271 276, June 2011.

[64] A. Picu and Thrasyvoulos Spyropoulos. Forecasting DTN performance under heterogeneous mobility: The case of limited replication. In *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, pages 569 577, June 2012.

[65] A. Picu, Thrasyvoulos Spyropoulos, and T. Hossmann. An analysis of the information spreading delay in heterogeneous mobility DTNs. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1 10, June 2012.

[66] Shuang Qin, Gang Feng, and Yide Zhang. How the Contact-Probing Mechanism Affects the Transmission Capacity of Delay-Tolerant Networks. *IEEE Transactions on Vehicular Technology*, 60(4):1825 1834, May 2011.

[67] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[68] Thierry Rakotoarivelo, Maximilian Ott, Guillaume Jourjon, and Ivan Seskar. OMF: A Control and Management Framework for Networking Testbeds. *SIGOPS Oper. Syst. Rev.*, January 2010.

[69] 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, June 2013.

[70] Filippo Rebecchi, Marcelo Dias de Amorim, and Vania Conan. Flooding data in a cell: Is cellular multicast better than device-to-device communications? In

Proceedings of the 9th ACM MobiCom Workshop on Challenged Networks, CHANTS '14, pages 19 24, New York, NY, USA, 2014. ACM.

[71] Manoj R. Rege, Vlado Handziski, and Adam Wolisz. CrowdMeter: An Emulation Platform for Performance Evaluation of Crowd-sensing Applications. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, pages 1111 1122. ACM, 2013.

[72] G. Resta and P. Santi. A framework for routing performance analysis in delay tolerant networks with application to noncooperative networks. *Parallel and Distributed Systems, IEEE Transactions on*, 23(1):2 10, Jan 2012.

[73] N. Ristanovic, G. Theodorakopoulos, and J.-Y. Le Boudec. Traps and pitfalls of using contact traces in performance studies of opportunistic networks. In *IEEE INFOCOM 2012*, pages 1377 1385, March 2012.

[74] Lucile Sassatelli and M. Medard. Inter-session network coding in delay-tolerant networks under spray-and-wait routing. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2012 10th International Symposium on*, pages 103 110, May 2012.

[75] Sebastian Schildt, Johannes Morgenroth, Wolf-Bastian Pöttner, and Lars Wolf. IBR-DTN: A lightweight, modular and highly portable bundle protocol implementation. *Electronic Communications of the EASST*, 37:1 11, Jan 2011.

[76] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD dataset cambridge/haggle (v. 2009-05-29). Downloaded from http://crawdad.org/cambridge/haggle/20090529, May 2009.

[77] Keith Scott and Scott C. Burleigh. Bundle Protocol Specification. IRTF RFC 5050, October 2015.

[78] Thrasyvoulos Spyropoulos, K. Psounis, and C.S. Raghavendra. Efficient routing in intermittently connected mobile networks: The multiple-copy case. *Networking, IEEE/ACM Transactions on*, 16(1):77 90, Feb 2008.

[79] Thrasyvoulos Spyropoulos, K. Psounis, and C.S. Raghavendra. Efficient routing in intermittently connected mobile networks: The single-copy case. *Networking, IEEE/ACM Transactions on*, 16(1):63 76, Feb 2008.

[80] Thrasyvoulos Spyropoulos, T. Turletti, and K. Obraczka. Routing in delay-tolerant networks comprising heterogeneous node populations. *Mobile Computing, IEEE Transactions on*, 8(8):1132 1147, Aug 2009.

[81] Jing Su, A. Chin, A. Popivanova, A. Goel, and E. de Lara. User mobility for opportunistic ad-hoc networking. In *IEEE WMCSA 2004*, pages 41 50, December 2004.

[82] R. Subramanian and F. Fekri. Throughput analysis of delay tolerant networks with finite buffers. In *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, pages 790 795, Sept 2008.

[83] Feiselia Tan, Youmna Borghol, and Sebastien Ardon. EMO: A statistical encounter-based mobility model for simulating delay tolerant networks. In *IEEE WoWMoM 2008*, pages 1 8, June 2008.

[84] G.S. Thakur and A. Helmy. COBRA: A framework for the analysis of realistic mobility models. In *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 145 150, April 2013.

[85] G.S. Thakur, U. Kumar, A. Helmy, and Wei-jen Hsu. On the efficacy of mobility modeling for DTN evaluation: Analysis of encounter statistics and spatio-temporal preferences. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 510 515, July 2011.

[86] Leigh Torgerson, Scott C. Burleigh, Howard Weiss, Adrian J. Hooke, Kevin Fall, Dr. Vinton G. Cerf, Keith Scott, and Robert C. Durst. Delay-Tolerant Networking Architecture. IRTF RFC 4838, October 2015.

[87] P. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. Dias de Amorim, and J. Whitbeck. The accordion phenomenon: Analysis, characterization, and impact on DTN routing. In *IEEE INFOCOM 2009*, pages 1116 1124, April 2009.

[88] Amin Vahdat and David Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, 2000.

[89] P. Vieira, A. Costa, and J. Macedo. A Comparison of Opportunistic Connection Datasets. In *2012 Third International Conference on Emerging Intelligent Data and Web Technologies (EIDWT)*, pages 66 73, September 2012.

[90] Wei Wang, Vikram Srinivasan, and Mehul Motani. Adaptive Contact Probing Mechanisms for Delay Tolerant Applications. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, pages 230 241, New York, NY, USA, 2007. ACM.

[91] John Whitbeck, Vania Conan, and Marcelo Dias de Amorim. Critical analysis of encounter traces. In *Proceedings of the 2010 ACM Workshop on Wireless of the Students, by the Students, for the Students*, S3 '10, pages 29 32, 2010.

[92] John Whitbeck, Marcelo Dias de Amorim, Vania Conan, Mostafa Ammar, and Ellen Zegura. From encounters to plausible mobility. *Pervasive and Mobile Computing*, 7(2):206 222, April 2011.

[93] L. Wood, W. M. Eddy, and P. Holliday. A bundle of problems. In *2009 IEEE Aerospace conference*, pages 1 17, March 2009.

[94] Eiko Yoneki. The importance of data collection for modelling contact networks. In *Computational Science and Engineering, 2009. CSE 09. International Conference on*, volume 4, pages 940 943, Aug 2009.

[95] Hayoung Yoon, JongWon Kim, Maximilian Ott, and Thierry Rakotoarivelo. Mobility emulator for DTN and MANET applications. In *Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization*, WINTECH '09, pages 51 58, 2009.

[96] Marko Zec and Miljenko Mikuc. Operating system support for integrated network emulation in IMUNES. In *Workshop on Operating System and Architectural Support for the on demand IT Infrastructure (1; 2004)*, 2004.

[97] Xiaolan Zhang, Giovanni Neglia, Jim Kurose, and Don Towsley. Performance modeling of epidemic routing. *Computer Networks*, 51(10):2867 2891, July 2007.

[98] Zhenjing Zhang, Zhigang Jin, Huan Chen, Yantai Shu, and Chuandong Zhao. Design and Implementation of a Delay-Tolerant Network Emulator Based in QualNet Simulator. In *WiCom 09*, pages 1 4, September 2009.

[99] J. Zhou, Z. Ji, and R. Bagrodia. TWINE: A Hybrid Emulation Testbed for Wireless Networks and Applications. In *IEEE INFOCOM 2006*, April 2006.