Imperial College of London
Department of Computing

# User Behaviour and Security in Opportunistic Networks

**BSc Thesis**

Natalia Krystyna Kulesza

Supervisor
**Dr. Emil Lupu**

Second marker
**Dr. Alessandra Russo**

## Abstract

This thesis investigates by means of simulation the effects of uncooperative user behaviour in opportunistic dissemination networks, in which a message originator can specify that the message should only be delivered to a trusted subset of the total node population. The nodes in the network are required to disseminate messages only to nodes whose membership in this desired subset they have confirmed, which is determined by a computational trust value that is assigned to each node. It is likely that some nodes in the opportunistic network will ignore this selection requirement, and therefore jeopardise the security of the network from the point of view of the message originator.

The investigations show the extent to which uncooperative dissemination behaviour negatively affects computational trust schemes that are applied as a security measure in opportunistic networks. Subsequently, a scheme is devised that helps to detect and exclude uncooperative disseminators from the networks, and conclusions are drawn about its effectivity and the conditions under which such a scheme can work.

A second contribution of this work is the creation of a mobility trace that is superior to existing mobility traces in terms of the realistic modelling of human mobility. This trace is used as an input to the opportunistic network simulations, and by being realistic allows to draw authoritative conclusions from these.

## Acknowledgements

I would like to thank my supervisor Dr. Emil Lupu, who not only helped me with continuous advice, support and motivation while I worked on this thesis, but also gave me deep insights in the ways of scientific working in general, which significantly increased the value this work has for me in terms of personal development.

I owe thanks to my second marker Dr. Alessandra Russo, for always having time for me and giving me kind and supportive help whenever I needed it. I thank you for being a fantastic tutor and for making me feel that I'm safe and have someone to help me from the moment I came to university.

Thanks to Hubert, for being supportive, kind and understanding, and for going with me through all those difficult moments with a smile.

Thanks to Anton for being the LaTeX-expert, and for being the best flatmate ever.

Thanks to my family for teaching me that nothing is impossible and for showing me what really matters in life. Thanks to you I can be where I am and look into the future with optimism and hope.

*Dla Hubika*

# Contents

# Chapter 1

# Introduction

## 1.1 Short Introduction to Opportunistic Networks

For many years now, there has been a clear paradigm shift from stationary to mobile computing. Starting with simple mobile phones, equipping them with more computational power, a memory, and wireless Internet connectivity has given a completely new experience and possibilities to the user. Modern mobile devices are even more than that, being a hybrid between a PDA, a mobile phone, a camera and a video player (not to mention the capabilities of the ubiquitous notebook).

A step that was particularly important from the point of view of the networking community was the equipping of mobile devices with short range connectivity. Looking back, mobile devices had been unaware of each other, their only connection being to the Internet or to a telephone network. The advent of bluetooth- and WiFi enabled devices created the potential of interconnecting them, thus turning these mobile devices into a network. This gave rise to a flavour of mobile networks called *Mobile Ad-Hoc Networks* (MANETs). The nodes in a MANET are mobile, so can change their location, apart from entering or leaving the network at will by switching on/off their connectivity. For instance, military troops that are equipped with short range mobile devices, and are connected to their base by a helicopter flying back and forth which carries information, could form such a network[1]. MANETs are continuously researched, e.g. by a dedicated group at the IETF[2], but also by university and industrial researchers. One of the major challenges in MANET research is routing, since the topology of the network constantly changes: A route that exists at one moment in time may not exists a few seconds later, since some devices on the route may have moved

[1] MANETs in the military are the subject of active research, for instance see [5]

[2] http://www.ietf.org/html.charters/manet-charter.html

away. The solution in MANETs is usually to keep track of the current topology in a more or less sophisticated way, in order to be able to make informed routing decisions, as described for instance in [24].

But recently, the question has been asked whether it would be possible to make routing decisions on the basis of exclusively local information, i.e. by making no assumptions whatsoever about the current topology. This approach could allow for a much more ad-hoc way of networking by greatly simplifying routing. This is because locally available data are accessible much more quickly and cheaply, their amount is much smaller, and no topology information has to be sent exchangedF. As an additional benefit, resource efficient routing would also allow very simple devices to be incorporated into the network.

A new kind of network was hence invented, the *opportunistic network*, in short *oppnet*. In its most discussed form, it consists of mobile devices that are carried around by their owners during every day life activities, and connections are created whenever two devices are in physical proximity, as in fig. 1.1. Routing algorithms in oppnets are an attempt to compute the probability with which a device in the vicinity will bring a packet to be routed closer to the destination, taking into account its current direction and similar locally available data. This opportunistic kind of forwarding gave the network its name. It has been shown that oppnets are a feasible type of network in terms of routing in real life applications, so this is by no means a completely esoteric idea. As an example, in Lapland (northern regions of Scandinavia), an oppnet is deployed that interconnects the indigenous Sami people with each other and with the bigger cities in the region by using opportunistic forwarding between the reindeer herders, along with some fixed location connectivity hotspots[3].

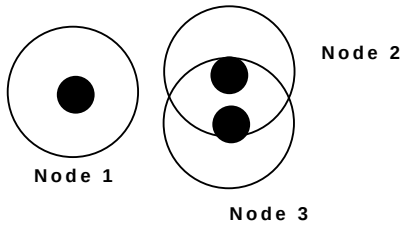


Figure 1.1: In this oppnet snapshot, Node 1 does not know of the existence of Node 2 or Node 3, since they are not within its communication range. Node 2 and Node 3 do know of each other and are able to communicate.

But as with any type of network, the actual deployment depends (among others) on appropriate security measures. This is particularly important since oppnets assume only wireless, relatively short range connections. No central

[3]See [2], for other applications see [24]

trusted authority can be used as a trusted third party because it won't be accessible locally. This is a considerable restriction: The lack of a central authority means that cryptographic signatures and certificates, many authentication protocols and encryption algorithms are not applicable.

## 1.2 Trust Based Opportunistic Data Dissemination

Among the solutions to security questions applicable in this situation, computational trust has been researched, with some encouraging results in other types of distributed systems[4]. A computational trust value is assumed to be very similar to the human notion of trust: Before a node in a network agrees to interact with another, it gathers information about the other node, and determines a trust value depending on previous interactions with this particular device or class of devices, on reputation values provided by third devices, and other application dependent data. This trust value can be applied to access control problems (is the device trusted enough for it to be allowed access to a resource?), to ensure confidentiality (can it see this piece of information?) and similarly to other security problems.

Trust in general has been researched since the early 70' [1], and applied to computing in the early 90' [37]. It is of course probabilistic in nature, so it is not applicable as the only security measure in high security systems, but it gives good results e.g. in multi-agent systems when used to significantly decrease the probability of a harmful interaction with a malicious agent [6]. And from there, it is a short way to applying trust to opportunistic networks in a similar fashion.

However, the application of trust values presents a unique challenge in the case of oppnets: In many trust models in the literature, trust relies to some extent on data that can be collected externally. E.g., in many trust models concerning P2P systems, a user, apart from assessing the trust into a file provider from his own experiences, can contact other users about their opinion of the provider, to assess the provider's reputation. If necessary, the user can wait for some amount of time for the opinions of other users to arrive. In the case of oppnets, all information that should be used must be locally and immediately available, considering that the durations of contacts are very short. Hence, reputation values in their pure form, a major source of trust information in other models, can't be used in oppnets.

Now there are many application areas in which trust can be applied in conjunction with oppnets. For example, one might think of a network in which everyone can dispatch a query ("Who currently sells a used car, model X, price up to Y?"), and wait for replies, this way sparing themselves the reading of many sale ads in newspapers, and possibly reaching previously unknown sellers. After

[4](see e.g. [36, 39] for P2P, or [6] for mobile agent systems)

getting replies, they would check the offers and rate the offers according to satisfaction for future reference for other users, and buy the best possible option. This is the standard model of providing and using recommendations in a trust framework, similar to that on eBay[5], and the components (dissemination, routing and trust metrics in oppnets) are relatively well researched.

However, the locality of communication gives rise to more a challenging problem. As mentioned before with respect to trusted authorities, interactions in such networks can't be observed by an entity outside the connectivity range of the communicating parties. This means that oppnets are inherently sensitive to user behaviour, since they rely on user devices behaving in a way that will benefit others, and this kind of good behaviour cannot be monitored or enforced directly. E.g. if a large set of nodes rely on other nodes to forward their messages, but are not willing to relay messages themselves, the routing network quickly breaks down. There has been work that investigates how egoistic behaviour affects the utility of an oppnet: In [12], an oppnet is set up in which each node has iHave and iWish lists (with the obvious semantics), and on each encounter they exchange data that the one has and the other wishes for. The authors investigate in how far it affects the system when there are nodes that only collect data from other devices, but don't share anything, and clearly this behaviour has been found to negatively influence the network utility. Even if the oppnet model used in the simulation is to some extent questionable, this finding is quite common in similar research papers such as [25].

Egoistic behaviour has to be taken seriously, as a study concerning a file sharing application found. A user behaviour study from 2005, [14], has collected data about the usage of the file sharing application Gnutella, and found that about 85% of users share no files at all, using the system egoistically only for down- and not for uploading, and that this number has increased from 66% in 2000. On the Internet, it is relatively easy to set a requirement that can counter egoistic behaviour, e.g. in Gnutella anyone wishing to download something may have to have an account with a minimal number of uploads. In oppnets, behaviour is not only not directly controllable, it is invisible to anyone who isn't situated in the nearest vicinity.

Imagine an oppnet which is designed to disseminate messages, with the constraint set by the message originator that they should only be given to entities with a trust value above a certain threshold. Is it possible to guarantee or at least encourage users to actually apply this trust value when disseminating the message, given that the trust value is computed on behalf of another entity, and not for the node's own security? In other words, if users are egoistic enough to use resources and not to provide any as in Gnutella, they are quite likely not to apply security constraints on behalf of other entities. In this thesis, I am fo-

[5] www.ebay.com

cusing on investigating various aspects of the effect of user behaviour on trust based security in oppnets, i.e. in how far this behaviour reduces the effectiveness of a trust value, and how to devise ways to counter this effect, or even provide control over dissemination behaviour that would encourage users to behave in a way which benefits the network. This is a challenging problem given that the message originator does not even know the nodes that the message will be passed through in an oppnet, and there is no way of controlling the transmission itself while it is ongoing. Even conceptually, it is more challenging than participation incentive schemes, since it is an attempt to make users behave in a particular way when participating, rather than just convincing them to passively make their resources available. On the other hand, it is an idea that would be very rewarding when implemented, since the possibility of encouraging users to apply a globally administered security measure would guarantee a significantly more stable and trusted system, while still retaining the ad hoc nature of the network with all its properties. With the inherent instability that is caused by user behaviour removed, an ad hoc network becomes easier to reason about, theoretical results become more applicable, and a practical deployment becomes more feasible.

## 1.3 Practical Application Scenario

In this thesis, I consider an example application of an oppnet where a global security measure is desired, and direct my efforts towards solving this more particular problem, to demonstrate feasibility and applicability and ease understanding.

**Problem:** Consider an event organiser advertising his events. It can be expected that he has a website where all relevant information is listed, together with the possibility of purchasing tickets. Typically, a person with a vivid interest in a particular kind of event will have a set of websites he checks regularly, or that he has subscribed to to send him latest event information. However meticulous his search, it is likely that he will miss events, particularly those of interest for a niche audience, and those that are not organised by a well known agency. It is in the interest of both our organiser and the customer to make the customer learn about our organiser's event, in the case of the organiser for financial reasons, and for the customer to feed his interest and broaden his horizon. But the organiser can't know everyone with an interest in his event, and the customer may have never heard of the organiser, so they have no means of contacting each other.

**Oppnet component:** Now assume that in the city they both live in, an oppnet is in operation which the people can subscribe to, and to which event organisers can feed their advertisements. To reduce undesired event information, the users specify their interests in a profile stored on their mobile device, and whenever they come in contact with a node in this network, they are transferred the latest information about current events in their interest area that this other

node has. This way, given a good dissemination mechanism, the event information can be spread throughout the entire city.

**Trust/Security component:** The organiser wants to ensure that the information is handed only to people who will pay for their ticket on time and behave appropriately during the event, to reduce costs associated with such misbehaviour and to increase the experience of "good" attendees. So each of the members of this network has installed an application on their device that can compute the trust of another device on the basis of its owner's previous behaviour at other events, and hand the message only to devices that meet certain trust thresholds.

**Behavioural component:** Ideally, if everyone applies this trust computation, undesired malicious nodes can be weeded out. However, it is quite likely that someone will eventually write a program that allows to ignore the trust value and disseminate the ad to anyone they meet, inclusive of potentially malicious people. The organiser will be interested in finding a way of tracking dissemination behaviour, and punishing users who do not apply the constraints to provide an incentive for good users.

## 1.4 Contributions of this Thesis

The aim of this thesis is to investigate an opportunistic network that uses computational trust to disseminate information to a trusted subset of nodes, to see how the behaviour of users who do not apply that trust algorithm affects the utility of the network. Subsequently, a dissemination control scheme countering the expected negative effect of uncooperative dissemination behaviour has been devised.

The investigation has taken the form of a simulation, since the inherent unpredictability in an opportunistic network cause it to be difficult to analyse analytically, and since a simulation allows for greater flexibility and adaptability in the choice of parameters and details of the implementation.

One of the inputs to a simulation concerning oppnets consisting of humans is a description of how these humans move. Before being able to focus on the message dissemination and trust issues, a technical problem had to be solved that concerns MANET simulations in general and stems from this need to model human mobility: Whether or not such simulation results are authoritative depends strongly on the assumptions about the way humans move, which is called the *mobility model* or the *mobility trace*. In chapter 2, a more thorough explanation of this topic is given. For now, it suffices to say that most existing mobility traces or models thereof are considered to be too unrealistic to yield results that are even approximately authoritative. I have therefore devised my own mobility trace, which is realistic given the state of the art, and which gives validity to the simulation results obtained using it as input.

I have created an appropriate trust computation formula and metric, and models of the user behaviour components that affect the network of event advertisement dissemination.

Subsequently, a set of appropriate simulations that allowed to determine the effects of user behaviour and trust schemes on the opportunistic dissemination network has been run. The effectiveness of the dissemination behaviour control scheme has been determined, together with any prerequisites that must be satisfied in order for such a scheme to work, and with any constraints that such schemes may be subjected to.

## 1.5 Organisation of the Thesis

In the following chapter, the background and related work relevant to this thesis are explained. In particular, opportunistic networks, mobility traces and computational trust are discussed. Chapter 3 describes the design decisions that affect the opportunistic dissemination network, most importantly how the trust scheme and the dissemination control scheme is designed, and how human behaviour pertaining to events and dissemination is modelled. The technical implementation is given in chapter 4. Chapter 5 contains the results of the simulations run to evaluate the dissemination control- and trust schemes, together with detailed comments and explanations of the results. Finally, chapter 6 summarises the most important contributions of the thesis and concludes with suggestions for future work.

# Chapter 2

# Background

This chapter summarises the related work relevant to this thesis. A brief summary of work concerning user behaviour in oppnets is given in section 2.1. Section 2.2 describes the characteristics and different types of opportunistic networks that can occur. The types of existing human mobility models are described and discussed in section 2.3, followed by a detailed account of formalisms, definitions and formulae relating to computational trust.

## 2.1 Related work

Simulations that show the effects of user behaviour in various flavours of distributed systems have been conducted before, and many of them follow a common pattern: A network is set up, nodes with different probabilities of behaving maliciously are introduced, the (always negative) effect of different degrees of malicious behaviour on some constituents of the network utility is measured and some counter-measure is proposed. Because of this similarity, these simulations are not discussed here for their own sake. The only concept that is interesting to take note of in these simulation descriptions is the distinction between different types of behaviour: Of the two that are relevant in this work, one concerns non-malicious, egoistic behaviour, such as using a resource without providing any in return, and the other actively malicious behaviour, such as accessing resources and knowingly damaging them.

The kind of behaviour that I concern myself with is misbehaviour at events in my event scenario, which is clearly active maliciousness, and dissemination behaviour. The latter is more difficult to categorise: You may argue that the nodes do not directly damage the event organiser, so they are passive, but on the other hand, assuming that the trust computation program is installed and

active by default for people first subscribing to our oppnet, the switching off of this program is in fact active maliciousness.

## 2.2 Opportunistic Networks

### 2.2.1 General Characteristics

In Opportunistic networks (*oppnets*), nodes are mobile devices with wireless networking capability, carried by their owners in every day life[1]. Hence, the nodes can change their location, and enter and leave the network at will, usually by means of the user switching on and off connectivity, or the entire device. A reader with knowledge of wireless networks will notice that this description fits a Mobile Ad-Hoc network, but the fundamental difference between those two lies in the knowledge of network nodes about the network as a whole: Nodes in Mobile Ad-Hoc networks (MANETs) keep track of the current network topology to make informed routing decisions, while nodes in oppnets make their decisions on the basis of knowledge available locally exclusively. The latter can't assume that a complete path to the destination node does or will ever exist, they can only forward packets to a node that is likely to bring the packet closer to its destination. This routing behaviour is called *opportunistic*, hence the name of the network class. Note that the lack of a path as explained does not mean that packets are never delivered, just that there is no *direct* path that the node can precompute when dispatching the packet. Rather, it means that the path does not exist at the moment, and if the packet is forwarded from node to node, one of them will eventually come in direct contact with the destination node.

This means that the packet delay depends on the quality of the chosen routing algorithm. Networks with this characteristic are called *Delay tolerant Networks* (DTNs). Many applications, such as email and data sharing tolerate delays in data transfer, so that MANETs are a useful alternative for many applications when network infrastructure doesn't exist or can't be used.

Depending exclusively on local interactions, oppnets are an interesting alternative to other networks including MANETs in the case where knowledge about the topology is not desirable to be assumed or not available at all. Additionally, in a non-opportunistic MANET, the requirement of knowing the topology means that the nodes' movement has to be minimal, otherwise updates about topology changes will take up an overly large proportion of the bandwidth; here, oppnets provide greater flexibility. Possible application areas that have been proposed are disaster management (as in [27]) where any other connectivity and topology may

[1] Any set of mobile devices other than personal devices can form an oppnet, as long as it satisfies some characteristics specified later. Personal devices are mentioned here specifically because they are most relevant to this work's application area, and because this is indeed the most discussed form of oppnets.

break down, and introduction of connectivity to rural or other sparsely populated areas where installation of fixed networks is not economic (as in [2]).

### 2.2.2 Device types

Two types of oppnets can be distinguished, depending on the kinds of devices that they consists of. The first one is the most obvious and simple oppnet type, in which all devices are of similar built and have similar computational possibilities. This is the kind of network found in many practical experiments, where researchers have a custom device and software produced for their purposes, so all devices are in fact the same. These networks are primarily useful for information dissemination and routing, and for experimental purposes, since they are uniform and relatively simple conceptually, so differences in device built don't have to be taken into account when making inferences from experimental results.

Another conceivable form of oppnet consists of *any* type of device with networking capability, including mobile phones, PDAs and sensor nodes. This kind of network has a much richer set of applications, in particular resource sharing, which is what oppnets were initially envisaged for [23]. The aim is to create a network which can perform a wide set of actions on its own, e.g. capturing a picture by a sensor node, transmitting it by a mobile device using human or vehicular mobility to a desktop, and analysing it there, all that without human action.

For the purposes of this thesis, I will focus on the first kind, since the second requires a thorough understanding of the hardware of many mobile devices, and the conception of software that is suited for the largest possible number of them. This is clearly out of scope of this thesis. The principle of trust is the same in both these networks, so both can be used interchangeably for my purposes.

## 2.3 Human Mobility Patterns

To evaluate data dissemination that is based on human mobility as in the oppnet considered in this work, it is necessary to have either knowledge or a good estimate of the way humans move. By movement, all the displacements of a human during every part of his life that are relevant to the particular application are understood. Well known examples of areas with an interest in mobility patterns are Palaeontology, where the migrations of tribes are relevant, and the management of public spaces, where it is important to ensure that all points of interest are accessible, depending on the number of people who want to access them and the given topology.

Mobility patterns relevant to this thesis represent the paths a human takes during a typical day, e.g. from home to a shop, to work, to visit remote family and back home again. With an increasing interest in MANETs and similar concepts

of networks that can be implemented using human mobility, mobility patterns that realistically model human movement have become a major topic of research, which has not yet yielded a fully satisfactory solution.

There are three basic ways to simulate human movement at this level, as presented below.

### 2.3.1 Synthetic Patterns

[2]A straightforward approach to human mobility assumes approximation by some mathematical function that is suitably randomised and determines the relevant mobility factors like speed, location and path length. Below the main approaches to synthetic mobility models are described:

- **Random Mobility Models**: As the name suggests, these models assume that mobility of nodes is random and independent in all relevant aspects (speed, pause times, direction...). The most prominent example is the Random Way Point model, in which all of these factors are drawn from a uniform distribution, and which has been widely used due to its simplicity. The main point of criticism of this model is that it allows for sharp turns and sudden changes in velocity, which is unrealistic (for a thorough analysis of this and other point of criticism, see [38]). This model may be suitable for a set of particular applications, but is certainly not a comprehensive model of human mobility: Apart from the unrealistic sudden changes it introduces, it has been shown that human mobility is centered around a few locations that are visited regularly, while random models do not take this into account at all [29].

- **Models with temporal dependency**: To counter the occurrence of sharp turn and sudden changes in velocity, some models assume nodes to have a certain amount of "memory" that will allow to derive current movement from the movement during the last few time units. For instance, a node that has been moving very slowly may increase its speed by some fraction relative to its previous speed, so that in effect it will still be moving slowly. An example from this class of mobility patterns is the Gauss-Markov model. Although more realistic than random models, this class of models suffers from the lack of any spatial considerations, so that nodes can move in any direction. In reality however, humans have to move around obstacles such as buildings or trees, and are likely to depend on the movement of other people in their vicinity, such as in traffic jams or on pavements.

- **Models with spatial dependency**: This kind of model solves the problem of nodes whose movement is dependent on other nodes, e.g. in the

[2]The information in this section is taken from [4], and is well supplemented by e.g. [34, 17]

situation of a node that moves behind another one, and can only move as fast as the front node does. An example model is the Reference Point Group model: A general direction vector is assumed to represent the direction of the group. This vector could be for instance a group leader, or a direction agreed upon upfront, and it moves with the group, so that it continuously forms a centre of it. At each time unit, the members of the group are placed around that centre according to some probability distribution, without interfering with each other. This model is useful to show node interdependency (on a pavement, or in military platoons), but it is only as good as the direction vector, which has to be modelled very carefully taking into account the existing topology.

- **Models with geographical constraints**: Finally, there is a class of models that take into account that moving nodes are restricted by the topology, i.e. buildings, trees or lakes to avoid, or streets to drive on. For example, in the Pathway model, the topology is represented by a (possibly directed) graph, where vertices represent obstacles, and edges the paths between them. Each node chooses at random one destination, and moves to it along the shortest possible path. When it arrives, it pauses for a random time interval, and resumes its walk to a new random destination. Note that this model is very similar to Random Way Point, except it takes into account movement restrictions. Similarly to the spatial dependency model, this one is only as good as the input parameters, e.g. where nodes would realistically go to, and which paths they would take.

The criticism of the various synthetic patterns given in this section is understood to refer to the suitability for a pattern to simulate real human mobility; the criticism may not apply to other applications.

When combining some or all of these models to overcome their limitations, some good approximations to human mobility might be found. However, any synthetic model can only reflect the finite number of requirements and constraints that its creator can think of, so synthetic models are far away from modelling humans in a comprehensive way. At the moment, they are only useful for simulations with specific set ups and goals, usually not involving humans. It is therefore not advisable to use such traces for human mobility based data dissemination.

### 2.3.2 Real Mobility Traces

The insufficiency of synthetic models has led to attempts to capture mobility experimentally. The set up of all these experiments has the common core of equipping a set mobile nodes (humans, animals, vehicles) with mobile devices capable of measuring mobility data, the precise data depending on the goal of

the experiment. All such devices have wireless communication capabilities and memory to store experiment data before it can be retrieved.

### Measured/Measurable data

Mobility traces in most cases capture a combination of the following kind of data:

- **Location data**: The location of the nodes is recorded at periodical points in time. These data can either be collected as GPS coordinates that may optionally be later on turned into coordinates relative to an origin in the experiment space (e.g. in [16]), or as wireless sightings of a stationary device that serves as a location reference point (partially in [32]). This is the most fundamental and "obvious" kind of mobility data, but has the drawback of being dependent on the particular topology in which it has been recorded, e.g. it is not sensible to use location related mobility traces collected in London to reason about mobility in Paris. In other words, this kind of trace captures absolute locations in a fixed topology.

- **Encounter of other mobile devices**: The sighting of other nodes is recorded for each node, together with the time at which it occurred. This kind of data is useful if the nodes form a network, and it is desirable to determine which nodes are in communication range at which times, and if there are any patterns to this. Sightings are recorded by letting the mobile devices scan for others periodically, and record any devices in the vicinity. This approach also allows to estimate the length of each contact opportunity, by checking how many consecutive scans show the same device. In this case, the data is much less dependent on topology, since it only conveys relative locations. Sightings are the most useful form of mobility data for routing or dissemination simulations. They can if necessary be obtained by running a network simulation on location data.

- **Transmission data**: This kind of data records technology and protocol specific transmission information. Examples are bandwidth, or size of transmitted messages, and they refer to a precise kind of technology, e.g. Wifi or Bluetooth. This is not information directly related to mobility, but is useful in that it allows to make informed and realistic assumptions in simulations.

Many traces also collect data about other network characteristics, but these are highly experiment specific so will not be taken note of in this section.

### Usage Concerns of Real Mobility Traces

A major drawback concerning the usage of real mobility traces is their small currently available number, and in most cases their statistically not very significant

size. This is mainly a consequence of the large size and high cost of any statistically relevant experiment of this type. Another concern is that real traces may be specific to a certain location or application type, if the experiment set up is not made very general (which is difficult for small experiment sizes). Consequently, the application area of any intended simulation has to some extent to be bent to fit the available data, or a mixture of real and synthetic patterns or several real patterns has to be used, if the mobility pattern does not cover all needs of the application.

A database of real mobility traces has been created which aims at collecting all publicly available data sets, the webpage can be found at `http://crawdad.cs.dartmouth.edu/index.php`. Despite the already great number of traces collected there, the database is currently nothing more than a repository, since the integration of several mobility traces into larger and thus more comprehensive and realistic ones is currently difficult because of the different formats that are used for each trace. Additionally, each of these traces is on its own application specific, so even combining several into a larger trace creates a trace that is not generally applicable.

However, real mobility patterns are still sought for because they are the best means to comprehensively simulate the behaviour of the nodes that are measured, so in the case where not only particular aspects, but the entire picture of mobility is needed, they are the best alternative. As an aside, they overcome the disadvantage of synthetic patterns that require a lot of mathematical insight to find an appropriate pattern even before it can be used. Real patterns are provided "ready to use", which saves a lot of time and effort.

### 2.3.3 Statistical Data

The third possibility to obtain human mobility patterns comes from the exploitation of statistical data concerning every day life movements. For instance, [17] describes how statistical data about US citizens such as the distribution of arrival times at work or of the duration of the average working day allows to model human mobility comprehensively and realistically (in urban environments). There are surprisingly few such mobility traces, despite their advantages relative easy of implementation.

In order to compute a trace of contact patterns from such a set of individual mobility data, a few more steps are required though. For instance, it is necessary to create a map (either real or modeled) that has streets and buildings. Each human should be assigned one of the buildings (in contrast to e.g. the middle of a lake) as a home and an office. Subsequently, the path from the current location to the next scheduled one should be computed on the basis of the street map, and encounters of two humans (or their proximity within a specified distance) should be recorded as a contact.

[9] is another work that creates a simulation in that way. The authors divide the movement of a human into different submodels such as arrival time at work and the choice of transport between locations, and create a high-level model that governs these. Some of the submodels are based on intuition, some on the work of others, so that in total, their model is less credible than that in [17]. However, they validate their model against two real mobility traces in terms of contact and inter-contact times, the standard characteristics used for comparison of mobility traces. Their trace is much more similar to the real traces than to Random Way Point, which is consulted for comparison, which clearly speaks in favour of statistics based traces, particularly when the statistics have a firm background.

The advantages of this kind of trace are that first of all, it is generic in that the traces are not dependent on an experiment set up or location, and that secondly it can be created as needed, with appropriate size and any desired map, thus overcoming the weaknesses of real traces. Being based on realistic data, it is also comprehensive, therefore better than synthetic traces. It is of course not real, but if it can be validated against a real mobility trace to determine that it is realistic by comparing descriptive mobility characteristics, it can be a valuable replacement for a real pattern. Given these advantages, I have decided to use statistical patterns, the one from [17] to be precise, as a basis of my thesis. I give a thorough description of my pattern in the fourth chapter.

## 2.4 Computational Trust

Computational trust is a security concept alternative to cryptography, certificates and digital signatures, in that it doesn't need a trusted authority. Essentially, it allows nodes in a network to predict the future behaviour of another node based on the previous experience with that node, and thus make an informed decision of whether or not to interact with that node again. Other motivations for trust based security frameworks include the appreciation of the dynamic nature of trust in contrast to traditional security means, and the fact that it allows for a more fine-grained approach to security. For instance, instead of giving someone access to an entire set of sensitive resources, each resource can assigned a minimal trust level necessary to access it, the so called *trust threshold*, and access can be dynamically granted to each resource separately depending on the current trust level of the user. Both the users' trust and the resources' trust threshold can change over time (example from [25]). A major area where trust is of importance is e-commerce, and most of us know the basic reputation systems that are applied to encourage participation on sites like eBay[1]. The system applied there relies on reviews by other consumers, and the more the positive reviews outweigh the negative ones, the more likely we are (supposed) to trust a seller or buyer. This

[1] www.ebay.com

particular kind of trust is very prone to attacks such as collusions of several reviewers to give someone else a good or bad reputation, by writing fake reviews for him. Trust systems devised nowadays are much more complex than that, and there is theoretical work that allows for scientific verification and reasoning about trust.

### 2.4.1 Formalisation of Trust

Trust is a notion clear to humans since we use it daily in many situations from buying products to driving among a large number of other drivers, but it is very difficult to define in a way that can be commonly agreed on and used as a scientific basis of reasoning. In the following, I will denote the node which has to decide whether or not to trust another the *trustor*, and the node whose trust is to be determined the *trustee*.

This section describes the ways in which trust can be defined, and what its constituents and factors are. The precise computation of trust values is discussed in section 2.4.2.

#### Common basis

To start with notions that have found widespread acceptance[3], several kinds of trust that are distinguished are the following (see e.g. [28, 1, 37]):

- *direct trust*: The trust that is built up during direct interactions of the trustor and the trustee, reflecting the satisfaction in the interaction.
- *recommended trust*: An opinion of a node other than the trustor about the trustee that can be used to make a more informed decision. This is sometimes more intuitively called the *reputation*.
- *derived trust*: The trust value formed from any of the above by discounting them by a factor that estimates their quality. Quality can be defined in different ways, which some examples about trust derived from reputation show well: In [19](p.24), quality is equal to the trust in the recommender to be "knowledgeable" and "tell the truth", while in [39], the similarity of that recommendation to the own trust is considered. The former bases its discounting value on (an estimate of) the amount and significance of previous experience of the recommender, and on his honesty. The latter models the fact that different nodes may have different priorities and experiences

[3]There is no agreement as to which names should be given to which notions pertaining to trust, so for instance [31] calls 'confidence' what most other papers call 'trust', and in [39], 'confidence' denotes a very similar notion to what is elsewhere called 'reputation', which may lead to some confusion. I adapt the naming conventions that have appeared in the majority of materials used for this thesis.

influencing their trust, so that even truthful recommendations may not be suitable.

Another important notion is that of what I find most suitably expressed as the *object of trust*, i.e. what precisely a node is trusted to do or to know. In [37], a relatively early paper to propose the use of trust in conjunction with computer security in general distributed systems, the objects of trust are e.g. providing good quality cryptographic keys, or following a given protocol as specified. Another example can be found in [39], where P2P file sharing networks are considered. There, an object of trust are the provision of acceptable bandwidth or good file quality. Given my oppnet application as in the introduction, objects of trust would be a usage of event information for personal and non-harmful use only, and the conformity with a predefined dissemination protocol .

Having outlined this common basis, we are at a point where each research team starts finding its own definitions and ideas. They are in most cases closely related, but make all the difference in an actual implementation. I present here a set of most common notions that pertain to trust, to show the reader both the versatility and the intricacies of computational trust.

#### Definition of Trust

One of the central questions to computational trust, once having defined the possible objects of trust, is what precisely trust measures. This is highly application dependent. E.g., again in [39], where the application area is P2P sharing, trust measures the "[...] degree of satisfaction with the interactions". In the case of someone downloading files this is a mix of *competence* and *honesty* to provide quality files, and the *reliability* to provide them with acceptable bandwidth. It is these three factors that are most commonly mentioned in the definition of trust as assumed by researchers, in different combinations. Three examples:

- "Trust is an agent's belief in attributes such as reliability, honesty and competence of the trusted agent" [39]. Here, the authors do not event specify the precise trust factors, just give three examples, as explained above.

- [37] defines trust as the fact that "A believes that B will behave in a certain way - perform (or not) some action (or actions) in certain specific circumstances." This simple definition abstracts away all characteristics of the trustee or its behaviour, and only deals with the actual outcome of an interaction.

- "[Trust is] the positive expectation that an interaction partner will act benignly and cooperatively in situations where defecting would prove more profitable to itself" [31]. In this definition, we have honesty and reliability

(if cooperativity can be called similar to reliability), but there is also the idea that trust is only relevant where the trustee might profit from harming the trustor. This last factor should be handled with care since it only applies to trust concerning honesty. Depending on the application, trust may only concern competence for instance, in which case this definition does not apply.

At first sight, it may seem that the object of trust is very similar to the definition of trust, but this is not the case. By an object of trust, I mean *what* is a node trusted to do, this can be seen as an action. In contrast, the paragraph above explains *under which angle* this action is evaluated, which I called competence, truthfulness and reliability.

#### Confidence

Confidence essentially means the belief that the general trust value a node has computed for another is reliable, that it reflects an accurate prediction of the future behaviour of that node. Confidence could be low in the case when there is little data to base a trust computation on, or if it consists mainly of recommendations from untrusted sources.

#### Risk

A notion absolutely basic to trust is risk, since without risk, there is nothing to lose and no trust is necessary. However, it is not always explicitly included in trust metrics. Some metrics constrain themselves to measure benefits, e.g., as cited before, the trust metric in the P2P network in [31] measures satisfaction. But this only means that the risk is relatively low, in fact it is stated in this work that poor quality downloads may lead to annoyance, clearly not a very detrimental state of things[4]. So implicitly, risk is always taken into account, and I find it sensible to either include it in the metric, or, if it is small and does not vary, at least state this explicitly.

In metrics that do use risk (e.g. [28]), it is usually one of the factors that are multiplied together to obtain the overall trust value, so it is a discounting value (i.e., the larger the risk, the closer the discount is to 0). This accounts for situations where a node may not be trusted even if it does have a high trust value, simply because the risk involved in a certain interaction is unusually high.

#### Utility

Utility is another basic notion, since of all the things that can be at risk utility is a very common one. In other words, trust problems arise when someone has reason

[4]Although the authors probably should take into account the possibility of downloading a file infected with a virus as well, which makes downloading more risky at once.

to believe that utility can be decreased by an interaction. As a practical example considered in [25], in an e-commerce system, the lack of trust in vendors and buyers due to a large number of fraudulent interactions is likely to make users stop using it. The malicious nodes in this case are insiders of the system (in that they have an account they can always access), so an ordinary authentication server doesn't help. The authors argue that computing a trust value that includes a measure of utility can help find nodes that decrease utility, and exclude them from accessing the authentication server. They also use this trust value to give nodes differentiated access to resources after they are admitted, giving more fine-grained control over the system.

### Importance

While utility can be viewed as a measurable value that allows for rational decision making, Stephen Marsh [28], who wrote a well known trust thesis on trust formalisation, identifies a subjective trust factor that he calls *importance*. Importance is a measure of how significant an interaction is subjectively, and in his thesis Marsh uses it as a discounting factor of trust. This means that an interaction that has little utility but high importance will be treated more seriously than an interaction with both little utility and little importance. Marsh highlights an interesting conceptual problem pertaining in particular to importance: One can take two views. An interaction with high importance associated with it should either have a high trust threshold since it is important to get it done well, or a very low one, to increase the chances that it will get done, i.e. there are two contradicting extreme possibilities that can both be applied to solve the same problem. The decision about which of these two should happen is of course situation dependent (a time constrained node will choose the second option, a quality constrained one the first), but this example shows that most careful thought has to be put into designing computational trust algorithms.

In this thesis, the emphasis is on utility rather than importance, so this problem will not be further gone into.

### Time

A small but nonetheless important characteristic of trust is that it is considered to change with time. Usually, the precise change is decay of significance of old trust information. Whether or not one should take time into account is mostly application dependent, but it is certainly important to at least consider it because the semantics of changes in time very clearly are appropriate and can contribute to the meaningfulness of the trust evaluation.

#### Components

While the trust factors and metrics are application area specific, the structure of a program that manages trust in a network node always contains a certain set of components, as described in [1]:

- Information Gathering: A part of the functionality needs to be dedicated to the gathering of the information on the basis of which trust will subsequently be computed
- Information Sharing: In the case where reputation values are exchanged between nodes, this component is responsible for their dissemination to other nodes as required.
- Information Modeling: This component combines all the trust values that can be processed and are available into a single trust metric.
- Decision Making: Finally, this is the component that uses the metric to make a decision about the behaviour towards the trustee.

### 2.4.2 Trust Computation Algorithms

The variety of published trust metrics, as can be expected, reflects the variety of trust definitions and models. It is far from trivial to choose one of them for my application area, or even to derive a custom made metric from them, since oppnets pose restrictions that other typical application areas don't have.

In particular, many trust frameworks assume either continuous [28] or on-demand [36] recommendation exchange, which is not possible in oppnets (as discussed briefly in the introduction, the physical locality of communication in oppnets makes only a very restricted set of potential recommenders available, if any at all).

The following gives an overview of some important considerations pertaining to the design of trust metrics.

#### Output

A definition of the format of the final trust value is quite intuitive, and almost uniform throughout the relevant literature. Consider the effect of trust, which is always a decision. One extreme case is blind trust, in which any interaction is committed to by the trustor, and the other extreme is absolute distrust, where no however safe interaction comes into life. It is intuitive to call these extreme cases 1 and 0, and to let all intermediate values be real numbers in the interval [0,1] This also models the probabilistic nature of the trust value.

There is one way to extend this interval, which e.g. [28] argues for, namely the widening of it to [-1,1]. The argument for such an extension is that there is

not only trust and a lack of trust, but also *distrust*. Semantically, where trust is an affirmative, encouraging value, distrust is an inhibitor. The number 0 in this case models indifference, or a lack of information[5].

Other outputs than a single number are conceivable as well[6]. Some trust metrics return a vector, which may be useful for the internal computations of an application, but as [20] argues, is not intuitive for humans, since it requires a complex evaluation to determine a decision from a vector, in contrast to real numbers which only need to be compared against a threshold value. An alternative that is of little use is a binary output, 1 meaning "trust", and 0 meaning "distrust" (or "lack of information"). While being intuitive, this value is not expressive at all, and furthermore inaccurate, since it merges the complex notion of trust with the final trust-based decision. While the decision is certainly binary since it measures whether or not the trust value is smaller than the thrust threshold, making the trust value itself binary is clearly a gross simplification.

### Trust combination

The way in which separate trust factors are combined into one meaningful output value is the defining point of any trust metric. It determines how generally applicable, and complex a metric is. However vast the number of proposed metrics is, most of them are based on a small number of trust combination techniques[1]:

**Discount values**: This is a value that regulates, or scales, the total trust by multiplication, in the case of real-valued output. Depending on its semantics, it can be multiplied with the total, or one of the constituent trust values, acting as an inhibitor to interactions. Good candidates for discount values are risk, importance and confidence.

**Means, Modes, etc.**: A simple way to combine several trust values into one is adding them and dividing by their total number, possibly giving each value a weighting. Other ways such as modes or medians are also conceivable. Together with discounting, this gives a number of possibilities for very lightweight and intuitive trust metrics, as e.g. in [25]. For instance, one could add several risk values together and add them by their total number to get a normalised, average risk factor.

**Bayesian statistics**: The simplicity of the above two methods has the disadvantage of being highly subjective, application dependent and potentially mathe-

[5] It is of course trivial to normalise such a range to e.g. [-100,100] or [0,100] respectively, or even to [0,∞], whichever seems most appropriate and intuitive to the application.

[6] see http://trustcomp.org/trustcomp-trust-values.html or a fairly extensive list

[1] The following categorisation is my own, on the basis of the trust metrics and algorithms I have come across while researching for this thesis.

matically not well-founded. A more objective and well-founded approach is given by *Bayesian inference*. It is based on the simple and well-known Bayes' Formula

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)} \tag{2.1}$$

which denotes the conditional probability of one event (A) given another (B). Now let the event B denote "all previous, relevant information concerning the trustee", and A denote "success of the current interaction". Then, intuitively, Bayes' formula computes a prediction of the future behaviour of the trustee, which is precisely the desired trust value.

But the idea of conditional probability still has to be incorporated into some kind of combination algorithm, and depending on the application, there are many ways to achieve this. One possibility is presented in [30], where the Bayes' Formula is used without modification or additional semantics, e.g. to compute

$$\Pr(\text{direct interaction experience} \mid \text{direct trust}) \tag{2.2}$$

to evaluate whether or not to interact with the trustee.

Other authors advocate the use of **Bayesian networks** (BNs), e.g. [39]. A BN is a directed, acyclic graph. The nodes are variables with a predefined set of values, and the arcs represent probability matrices that give the probability of a child node being in a particular state given that its parent is in a particular state. [39] uses BNs as follows, in the context of P2P file sharing:

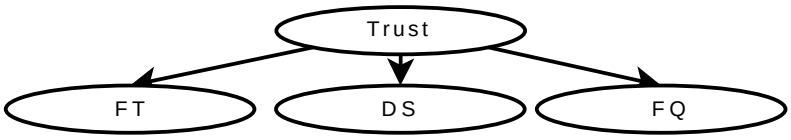


Figure 2.1: Simple trust BN in P2P context

Here, $FT$ stands for file type, $DS$ for download speed, and $FQ$ for file quality. A BN of this form (a root with leaves, no intermediate nodes) is called *naïve*. Semantically, the root node in such a BN represents a cause, and the leaves are the consequences. So the above network says that if someone is trustworthy, then he is likely to provide, among others, high quality files of a particular type. The arcs represent the conditional probability with which this happens, so that the arc between "Trust" and "FT" represents the *conditional probability matrix* (CTP) represented in table 2.1.

The first column represents the set of states of the child node "file type", and the first row the states of the parent node. Now given that we have enough experience to estimate the probabilities above, and the probabilities $\Pr(T = 1)$ and $\Pr(FT = "Music")$, Bayes' Rule allows us to compute the *reverse* conditional

| FT | T = 1 | T = 0 |
|---|---|---|
| Music | Pr(FT = "Music"$\|T = 1$) | Pr(FT = "Music"$\|T = 0$) |
| Movie | Pr(FT = "Movie"$\|T = 1$) | Pr(FT = "Movie"$\|T = 0$) |
| Document | Pr(FT = "Document"$\|T = 1$) | Pr(FT = "Document"$\|T = 0$) |

Table 2.1: Excerpt from example CPT in [39]

probability, namely Pr($T = 1$|FT = "Music"), "if I download music from that file provider, how likely is it that he will prove trustworthy?". This is where it becomes clear how useful a BN can be. All that is necessary to predict the future behaviour of an entity is the total trust Pr($T$), the probability that I will want to download music, and a conditional probability table that can be derived from experience.

Other, much more complex and expressive forms of BNs can be conceived, which allow to take into account many more and much richer interdependencies. For more detailed explanations, see [15]. BNs have the advantage of being able to cope with a vast number of variables, of being representable in human-readable form rather than a as set of simultaneous equations, and of being very well researched. They are in addition computationally very efficient, since computations involving them only need to perform matrix operations on the CPTs.

**Subjective logic** is another way of probabilistic inference. The logic model is described in [19], while the mathematical basis, the so called *Beta Reputation System* (BRS), is given in [20][7]. The author intends to publish a book that will include both of them, together with more recent material, in one publication, until now a draft version has appeared [18][8].

The logic itself is certainly interesting as a rigorous formalisation of beliefs, but is not relevant here.

The mathematical basis of the BRS is the Beta probability density function,

$$Beta(\alpha, \beta) = f(p|\alpha, \beta) = \frac{\Gamma(\alpha) + \beta}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \tag{2.3}$$

This function is given as input the number of times that two mutually exclusive and exhaustive events (say $x$ and $y$) have occurred, denoted $\alpha - 1$ and $\beta - 1$ respectively. It produces a *pdf* of the probability p of getting the event $x$ in the future, given the previous ratio of $x$ and $y$. Once again, it is not the condi-

[7]The mathematics is described in [19] as well, but I cite [20] separately because it is a publication dedicated exclusively to this topic.

[8]See also the interactive applets that were created for demonstration purposes, available on http://persons.unik.no//josang/sl/.

tional probability p of that event that is returned, but the pdf of that probability occurring. The mean of this distribution is

$$E(p) = \frac{\alpha}{\alpha + \beta} \tag{2.4}$$

Interpreted semantically, this means that the probability of the event $x$ is uncertain, but it is most likely to be $E(p) = \frac{\alpha}{\alpha+\beta}$.

One could use this directly to infer future behaviour from previous behaviour counts (e.g., estimate the probability of "x = good" vs. "y = bad" behaviour). A modification proposed by the author to make the BRS more suitable to express trust is to let r denote a real-valued *satisfaction* degree, and s a *dissatisfaction* degree. Subsequently, the original function can be used with $\alpha$ = r+1, and $\beta$= s+1, to give appropriate semantics. This new distribution is now called $\varphi(p|r, s)$

The return value lies in the interval [0,1], but since the author advocates an interval of [-1,1], the mean of that new distribution is scaled to give what he calls a *reputation value*,

$$Rep_T^X(r_T^X, s_T^X) = [E(\varphi(p|r_T^X, s_T^X)) - 0.5] \cdot 2 = \frac{r_T^X - s_T^X}{r_T^X + s_T^X + 2} \tag{2.5}$$

where the superscript denotes the trustor, and the subscript the trustee.

Trust combinations from several sources, say from $X$ and $Y$, in this model amounts to an addition of satisfaction and dissatisfaction values from each contributing trust value into $r_T^{XY}$ and $s_T^{XY}$ respectively, and using this as input to $\varphi$ as shown before (unless the reputations should be weighted differently, in which case another function is used).

Other operations in this framework are similarly simple, and can be looked up in the original paper.

#### Initialisation of trust thresholds

Initialisation of trust thresholds is perhaps the least mentioned problem throughout the literature considered for this thesis. Although some authors who implement example trust frameworks sometimes state the value they used (e.g. [25] use 0.5 $\in$[0,1]), these values are generally considered to be application dependent, and there seems to be no significant theoretical work that would help in the setting of these values. They have to be set either according to some subjective and more or less arbitrary idea, or determined after a set of experiments. Marsh in [28] does determine a formula for the "cooperation threshold", which is a function of the subjective values of risk, the trustee's competence, the initial trust and importance, but he points out that his idea poses some problems that he leaves for later work.

#### Initialisation of trust in nodes

A major problem is the question how to initialise trust before any interactions have taken place. A simple idea would be to let it be the half of the total trust interval, i.e. 0.5 for [0,1] and 0 for [-1,1], since these values represent indifference or lack of knowledge. This is however naive, since it is possible that all trust thresholds are higher than that, in which case the system is deadlocked.

Clearly, we need the threshold of at least one interaction (that will certainly take place) to be smaller than the initial trust value of a node interested in it. And for the practical purposes of a simulation, a smaller trust value leads to less interactions, so the system takes a long warm up time. On the other hand, initial trust should not be too high, since this increases the probability of damage. In the end, this is a decision of the authors of each simulation, and is not very well researched. E.g. the authors of [25] assume the initial trust to be just slightly higher than the trust threshold (there is only one general threshold in their simulation, and it is 0.5), but admit that this is a value based on intuition, and list better research for initialisation values as part of future work objectives.

### 2.4.3 Applicability of Trust Metrics in Oppnets

There have been findings that could have made trust metrics in oppnets infeasible by principle ([26]). Fortunately, it turned out that these findings were not as restrictive as they seemed. The problem is described here since some papers are still based on the assumption that the problem exists. The two notions necessary to understand the problem to be presented are *contact times* and *inter-contact times* of nodes in a network. The two mean, quite straightforwardly, the length in time units of a single contact opportunity, and the length in time units between two contact opportunities respectively.

It has been found that inter-contact times of nodes in DTNs follow a power law [26]. The consequence of this fact would be that the probability of the occurrence of *infinite* inter-contact times is quite significant. That in turn makes trust values hard to apply, because it is quite possible that a node once encountered, will never be met again. It is of course still possible to base trust values on particular characteristics that a group of nodes may have, in which case a node's specific identity has less meaning, but it is a fact that complicates things[9].

Fortunately, this finding was soon revised [22] by more thorough analysis of the available traces, and the new suggestion was that inter-contact times in fact follow a power-law distribution, but only up to a certain point, the very tail of the distribution being exponential, which removes infinite inter-contact times.

[9] Actually, the inter-contact time distribution has been studied with routing in mind, but of course the findings are equally applicable to computational trust considerations

Only one month later, even this was revised: in [7], the authors find that the previous papers consider *aggregated* inter-contact times instead of the inter-contact times of a particular pair of nodes, which is of course the data of interest. They find that pair-wise inter-contact times are better approximated by log-normal or exponential curves, and furthermore that the power-law distribution of aggregated data can be analytically derived from pair-wise inter-contact times.

This latter finding finally removes doubts about the applicability of trust to oppnets on the basis of long inter-contact times.

## 2.5 Conclusions

The background provided in this chapter has explained, additionally to giving an introduction to opportunistic networking, the state of the art in mobility model research, and highlighted the facts that so far, there is no fully satisfactory solution to the problem of modelling human daytime mobility. Work on a statistical mobility pattern has then been presented that can give rise to a realistic and flexible mobility trace, thus overcoming the problems of real and synthetic mobility traces. This work will be used and enhanced by me to create a mobility trace that surpasses the existing ones in terms of realism and general applicability.

Secondly, after an introduction to the concept of computational trust, possible approaches to designing a trust scheme have been described. Trust is a largely subjective concept, and therefore the accuracy and applicability of each such approach has to be determined on a case-by-case basis. For the case of a message dissemination network, I will use a mixture of the Beta Reputation System (BRS) and scaling factors. The BRS, due to its being mathematically well founded and being inherently capable of prediction, will be used to determine a predicted behaviour value from data describing the previous behaviours of each node, and this value will be scaled to meet the subjective needs of the organiser.

# Chapter 3

# Model Design

In this chapter, all important design decisions pertaining to the opportunistic dissemination network envisaged.

The first section gives a detailed model outline of this network. Subsequently, the trust formula that will be used to determine the nodes' mutual trust is presented, by first introducing the form in which trust data about each individual are collected, and then describing the formula that is used to determine the final trust value. These sections also gives the dissemination tracking scheme that allows to detect and punish uncooperative dissemination behaviour, and the overall protocol according to which any two nodes in the network communicate.

Then, I present the relevant models of behaviour of network nodes with respect to events and message dissemination, and mention security considerations relevant to the network as it is described.

## 3.1 Model Outline

To make it easier to follow the detailed explanations of design decisions, I give a coarse summary of the oppnet in the application scenario we are considering.

The purpose of the network in the application model is to opportunistically spread event advertisements to all subscribers who are both interested in it and meet a certain trust threshold. Subscription is done by registering and obtaining a piece of software that can compute the trust value with a predefined trust computation algorithm for each encountered device.

An event organiser defines an advertisement with the desired content, and includes a trust threshold in it for reference for the trust algorithm. He then dispatches the message to the network.

From then on, the message is passed from user device to user device (by user I mean a subscriber to this network). To prevent users from receiving advertisements of events they are not interested in, and to reduce the number of messages they have to store, each user can define a profile of interests where he can specify the kind of event he would like to be notified about, and only advertisements of interest will be shown to him[1].

Each time a message is to be passed to a user's device, the potential recipient's trust value is computed, to make sure that he has the right to receive it. The data about this individual that allow to determine his trust are stored on the recipient's device, and are signed digitally by the organiser, who is the originator of these data. The current holder of the message can verify that they have not been altered by using a verification key that is disseminated as part of the message. These trust data record the previous behaviour of this device's owner at events, so that the overall trust value is a prediction of his future behaviour. If it is high enough, the recipient is given the message, and can use it to inform himself about events if interested, and later redisseminate it.

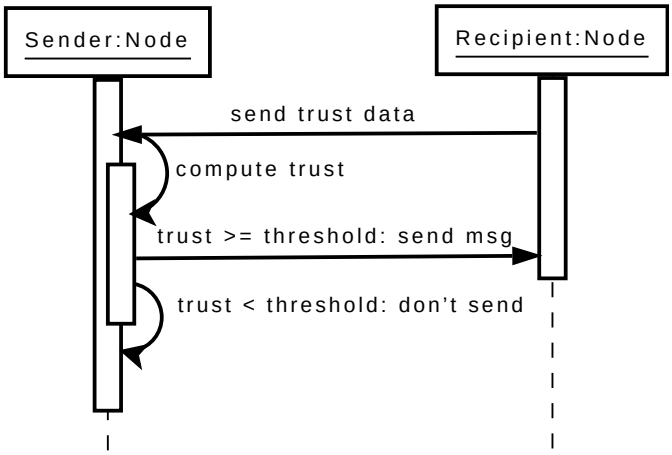


Figure 3.1: The transmission of a message to an interested node in pseudo-UML

After an event, each user device is updated with a log of the current event directly by the organiser just before the user leaves the event venue. This log consists of the new trust data that reflect how the user has behaved at this event. This will be taken into account next time the trust value is computed. The new, total event log is singed to prevent the user from altering it to his advantage. Note that there may be many organisers of the same kind of event. If they all have the same signature key, and update the same log, they are able to access a

[1] An uninterested user does receive the message, however it is not stored on his device permanently but deleted after re-dissemination, and not given to him again. This speeds up the dissemination process without affecting uninterested nodes too much.

larger set of behaviour data from many event logs, and their predictions become more accurate.

The aim of the organiser is to be able to accurately predict the behaviour of any possible event attendee based on an appropriate number of previous experiences, so that the event advertisement will only be given to users who are expected not to misbehave. The basic prerequisite for this scheme to work is the periodic access of the organiser to the user devices, allowing individual trust data to be updated, and this is given by the physical attendance at events. Diagrammatically, the trust evolution can be illustrated as in fig. 3.2.

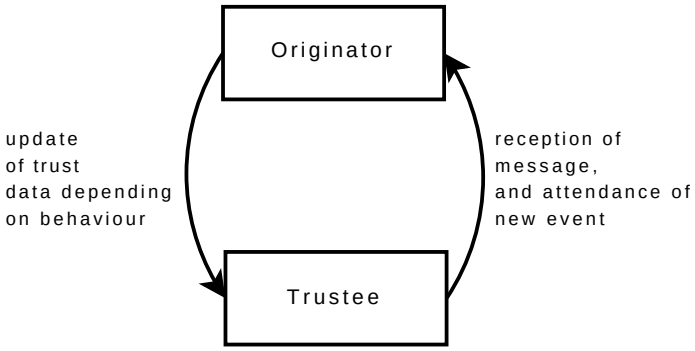


Figure 3.2: Abstract model of trust evolution

The issue investigated in this thesis is that the user devices may or may not disseminate the message correctly, e.g. it is possible to install an application on the device that will disseminate messages without the trust comparison between user trust and trust threshold, so that the trust update cycle is incorrect. I will measure the extent to which dissemination behaviour affects the correct functioning of the trust update cycle, and devise a mechanism that will help to alleviate any negative effects this may have.

The implementation of this network requires a trust computation formula, the mathematical formalism that allows to turn the trust data into a single trust value, furthermore a communication protocol according to which nodes exchange their messages (the essential part of which being the trust computation and subsequent message exchange), and a data structure that will allow to track dissemination behaviour. All of these are described in the subsequent sections.

## 3.2 Input to the Trust Computation Formula

Before presenting specific design choices, it is important to give a brief explanation of the nature of the trust algorithm. In the case of event dissemination, like in many other conceivable information dissemination scenarios, a judgement of

whether or not to give someone a message is based on objective observation of that person's behaviour. Hence the individual trust data that are collect for each user are quantitative observations of behaviour, defined in a way that is unambiguous from the point of view of any node in the network. The purpose of the trust algorithm is to extract a prediction from these that is as objective as possible, and then scale it by subjective factors to bend it to fit the current needs.

The algorithm takes as input two types of data: The trust data that pertain to the particular user whose trust is currently computed, and some parameters such as how much weight should be assigned to misbehaviours when they occur.

### 3.2.1 Individual Trust Data

Each node stores its own individual trust data. The decision to store the trust data of each individual on his own device means that the network scales to any size. Additionally, it would be difficult to keep other nodes' trust data up to date, since at any moment these nodes may attend events and therefore obtain new trust data that most other nodes in the network won't know about immediately. It consist of a log of all events attended previously. Each log entry contains an indication of the user's behaviour at that event. This may either be positive, or negative with a specification of the precise misbehaviour. Such data allow to infer

- the number $n$ of times a device has attended an event of a particular type in total
- the number $m$ of times a node has behaved as expected, i.e. non-maliciously
- for each type of misbehaviour, the number of times it has occurred ($mis_1$, $mis_2$ etc.)

These logs contain all relevant information about a particular person's behaviour, and can be used to obtain an objective estimate of the future behaviour of that entity.

For simplicity, I assume that there are three categories of misbehaviour, which can be called *serious* (e.g. breaking organiser's property), *moderate* (e.g. being inappropriately drunken) and *slight* misbehaviour (e.g. cancelling tickets at short notice). Depending on the type of event, these categories can contain different types of misbehaviours. Each of these categories has a different *utility loss* assigned to it, which indicates how much the trust value should be lowered if such misbehaviour occurs.

So a log entry for a particular event is either *positive*, *slightly misbehaving*, *moderately misbehaving* or *severely misbehaving*. This kind of scheme, in addition

to containing all information that is needed, allows to accumulate local trust data (i.e. frequency counts) by adding them together. Given one count of total event attendances and of good behaviour respectively, and three misbehaviour counts, this amounts to a total of five integers that form the local trust data. This simplicity can be a step towards using trust in mixed-device oppnets (see 2.2.2), since even computationally weak devices should be able to store and compute all information necessary to use this trust formula.

### 3.2.2 Global Parameters Set by the Organiser

The data in the last section are those that pertain to a particular user, and need to be complemented by data concerning the trust computation formula itself. The latter are disseminated with each message to allow the organiser to change them (e.g. if a change in the network or another reason cause the organiser to reconsider how he evaluates the trust data), and represent input parameters to the computation, set by each organiser individually. They are

- the trust discount (or: *utility loss*) $ul_{mis_i}$ which is associated with each misbehaviour $i$.
- initial trust values $t_{init}$ given to nodes that have never attended an event before, and thus do not have previous trust data
- the trust threshold $t_{thr}$associated with the message
- a verification key to verify the integrity of the individual trust data

After accessing the device specific data, the global parameters are used to combine the former into a meaningful trust value by the trust computation algorithm, which is stored in each device that has subscribed to the network.

With respect to the brief explanation of my trust algorithm given at the beginning of this section, the individual trust data are the objective values that help to make accurate predictions, and the global trust data are the weightings that allow to give a subjective meaning to them.

## 3.3 Trust Computation Formula

Having an overview of the data according to which trust is computed, it is necessary to define a function that combines these into a meaningful and representative trust value.

Fist of all, in the case of nodes that don't have previous trust values because they have never been to an event before, the initial trust value $t_{init}$ is used, which

is always higher than the trust threshold to allow nodes to enter the system initially.

To compute trust values, I use a mix of the techniques presented in the Background section. I believe that the objective part of the trust computation algorithm should be mathematically well-founded and robust, while the subjective scaling component is best represented by a simple and flexible discount value. For the objective component, I will use aspects of the Beta Reputation System (BRS) (see 2.4.2).

The overall trust value is computed according to the formula

$$T_{total} = Trust \times Risk \tag{3.1}$$

This equation, advocated for in [28] as well, means that some basic trust value, which is the prediction component, is discounted depending on the risk associated with the interaction. In fact, the trust formula has an additional confidence component, which additionally weights the trust value down if it is computed on the basis of only few trust data point. I incorporate confidence into the trust scheme in a different way, by assigning the initial trust value to any node that has fewer than four trust log entries. The trust entries are only used for trust computation after a minimally representative number of them has been collected.

In the formula 3.2, let $ul_j$ be the utility loss associated with misbehaviour $mis_j$, i.e. the perceived damage resulting from a particular misbehaviour. Here, $ul_j \in [0, 1]$, so that small utility loss could be mapped onto 0.2, and large utility loss onto 0.8 .
Let n be the total number of events attended.
Let $r_j = E(Beta(mis_j, n - mis_j))$, the mean of the Beta distribution which computes the probability of misbehaviour $j$ occurring, given all the behaviour data so far. The Risk factor in eq. 3.2 is then computed as:

$$Risk = 1 - \frac{r_1 \times (1 + ul_1) + ... + r_i \times (1 + ul_i)}{2i} \tag{3.2}$$

In essence, this formula first computes the probability that each misbehaviour will happen, and then multiplies it with the utility loss associated with this misbehaviour, which gives total risk for this particular abuse. Using $(1+ul_i)$ enhances each risk factor proportionally to its utility loss. These values are added together and the result is mapped onto the interval [0,1], giving an average measure for the total risk, and deduct it from one (for high risks, we want to multiply trust by a *low* value, to get low trust). Note that although $Risk$ is used as a discount value in the high level formula, it consists of a set of predictions, so the BRS is used here.

The basic trust component of eq. 3.2 is computed as

$$Trust = E(Beta(m, n - m)) \tag{3.3}$$

i.e. the probability that good behaviour will occur, given all previous behaviour.

To compute each probability of misbehaviour, the Beta distribution is parametrised with integers, and not real values, in contrast to what is explained in 2.4.2. The semantics of using real values is that each recommender can give a *degree* of satisfaction and of dissatisfaction as recommendation, but in the case of event dissemination it is preferable to use a more objective measure like quantity of misbehaviour, which is integer-valued, and allow each organiser to assess for himself how to discount the quantitative prediction given by the Beta distribution as is used here.

As a last point, to compute $Risk$, the two complimentary event counts given as input parameters to the Beta distribution correspond to the frequencies of a particular misbehaviour, and the total number of event attendances. I initially considered to give as a second argument the total number of *misbehaviours*, which would result in the probability of a particular misbehaviour *given that misbehaviour occurs*. Eventually I decided against this approach, since it does not reflect the absolute, but the relative probability of a misbehaviour occurring, which overweights risk in my formula.

### 3.3.1 Dissemination Tracking and Trust

The core part of this thesis is to find a mechanism that enables the organiser to track the dissemination behaviour and punish it by decreasing the trust value.

The difficulty in devising such a scheme lies in the fact that the entities who are on the path of a disseminated message are not necessarily those that attend events in the end. With attendance being required for trust updates, this means that a different approach to dissemination trust is needed than for event behaviour trust. There may be entities that choose to disseminate the messages of organiser A correctly, because it is A's type of events they are interested in, but who do not apply the trust algorithm to organiser B's messages, because they never attend them and so will never get punished.

A solution to the first problem comes when considering that in contrast to nodes communicating among each other, organisers are not restrained by the locality of communication at all. They can be assumed to have permanent access to the Internet, and some powerful workstations. They can *share* trust information. If one of them detects a user with bad dissemination behaviour, it can update a *globally visible* trust profile, and whichever event the user chooses to attend next, its organiser will be aware of the issue and will update the user's local trust profile at the event. Hence, the correctness of the trust data of an individual is independent of which events the user attends and which messages it disseminates how.

To integrate dissemination trust into the trust scheme seamlessly, the punishment of misbehaviour is implemented as a manipulation of the nodes' individual trust data, for instance by increasing the 'moderate misbehaviour' count.

The next question to answer is how to track dissemination behaviour, when communication is not directly supervisable. This can be achieved by letting each device add its own ID together with a timestamp to the forwarded message, so that a complete path of disseminators is available for each message. When a user attends an event, he hands his message to the organiser. The organiser then computes for each ID in the path what trust value it had at the timestamped point in time, from the (historical) global trust data. If the trust value is smaller than the trust threshold, this means that the previous node has disseminated the message incorrectly and should be punished. He adds this information to the global trust data, and whichever organiser sees the node next will update its local trust profile accordingly.

The reader will have realised that this scheme catches misbehaving disseminators by making them include their ID in a message before they forward it. Fortunately, the properties of oppnets make the signature scheme more powerful than it would be e.g. on the Internet: Consider a typical message. The first user appends his ID to the message, and signs it digitally. The second user appends his ID, and signs it *together with the previous signature*. This creates a chain of signatures depicted as below:

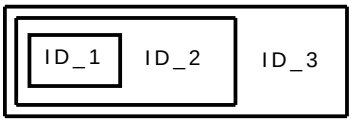


Figure 3.3: Nested signatures that allow for precise tracking of the path of a message. Boxes represent a signature of the content in it.

The dissemination protocol will then have to be updated to check for each incoming message whether the previous user has signed himself correctly [2], and if not, takes note of that. When he attends the next event, he will report any user violating the signature scheme. Now on the Internet, a malicious node could quite simply decide to disseminate the message only to nodes that he knows to be malicious as well, thereby making sure that it is not reported. In an oppnet however, it is not possible to determine in advance which nodes are going to be in the vicinity, so it is not possible to anticipate with whom communication will take place in the future. This is an incentive to give correct IDs, since chances are that one will eventually meet a non-malicious node and get reported (unless we assume

[2]The assumption here is that the real user ID is always visible in direct contact. Identifying network users in direct contact is a question of ongoing research and there is no comprehensive solution that I have heard or read about, see section 3.5.

that nearly all users are malicious disseminators). Similarly, the timestamps can at least roughly be verified by taking as reference points the directly preceding and succeeding entries.

The individual trust data have to be updated as well to contain a signature key used for creating the chain of signatures. The verification key needs only to be held by the organiser, since he is the only one who will check the message to process the dissemination data.

### 3.3.2 Summary of Communication Protocol

The full communication protocol can be summarised as follows:

```
if(trustor applies trust scheme)
    obtain trustees trust data
    if(trustee's trust data don't correspond to their signature)
        abort communication and remember trustee to report him
    compute trustee's trust value
    if(trust value < threshold)
        abort communication
    sign and timestamp message
    give trustee message
    trustee checks trustor's signature
    if(signature doesn't correspond to trustor's ID)
        remember trustor to report him
else
    sign and timestamp message
    give trustee message
    trustee checks trustor's signature
    if(signature doesn't correspond to trustor's ID)
        remember trustor to report him
```

## 3.4 Simulation of Users

### 3.4.1 Behaviour Model

To simulate a population of humans with differentiated behaviour pertaining to event behaviour and dissemination behaviour, a model needs to be devised that describes the relevant characteristics of these humans. These are likelihood and severity of event misbehaviour, interest or lack thereof in the advertised events, and whether or not a node has bad dissemination behaviour.

All the behaviours modelled here are represented as real numbers, between 1 and 0, with the exception of the "interested in the event" and "uncooperative

disseminator" characteristic, which are binary valued. Additionally, these behaviours do not change over time, since they are perceived as properties of a nodes' "personality".

### Event Behaviour

The first of three behaviour models that need to be defined for each user is the event behaviour. It consists of a decision on whether or not a user misbehaves at an event, and if so, how severe this misbehaviour is. This is implemented by giving each node a mean and standard deviation of event behaviour, the "personality" of that node so to speak, and during each event, a value is drawn from a normal distribution with these parameters. This value is compared against the utility losses associated with different misbehaviours, and choose the largest one that the behaviour value exceeds as the current behaviour of the node. To clarify, if the behaviour value is 0.7, and the utility loss of moderate misbehaviour is 0.6 and that of severe misbehaviour is 0.8, then that node is misbehaving moderately. If a node's behaviour does not exceed the "slight" utility loss, it is considered to behave well.

The mean described above is chosen from the normal distribution with parameters (0.1, 0.1), which intuitively says that the average event attendee will be well behaved - 80% of all nodes have a mean misbehaviour of less than 0.2, which is the lower limit of slight misbehaviour, i.e. they behave well. If misbehaviour occurs, it is with 99% probability slight misbehaviour, since 0.99 of the distribution lie below 0.4, the lower limit of moderate misbehaviour. The standard deviation is chosen to be twice the mean, to reflect that the someone who is likely to misbehave is in fact capable of any behaviour, while someone more well behaved will not vary much in what he does.

### Event Interest

It is also necessary to model which of the users are interested in the event at all, and which of this latter subset, having received an advertisement, actually attend the event.

As mentioned, interest is binary valued, and is used to specify which event ads a node wishes to receive by in/excluding the corresponding event types in its profile. I assume that an average of 10% of all nodes are interested in the event, given that they are active enough to participate in a network which encourages event attendance.

I need another value related to interest, which reflects how likely the user is to attend an event once he learns of it, i.e. how active and involved this node is given his event of interest. This will subsequently be called *interest degree*. The assumption made here is that the greater the interest-degree, the more likely the user is to actually come to the event; it can't be assumed that anyone who

gets an advertisement is equally likely to attend, so I want to allow for a real valued interest degree. Now what is necessary to attend an event is of course the purchase of a ticket (or simply registration, if it's a free event). There are finitely many tickets, so only early ticket purchases are successful. I use the interest value as follows to determine which nodes go to an event:

$$Time_{ticket} = Time_{current} + Time_{remaining} \times (1 - (Interestdegree)) \quad (3.4)$$

$Time_{ticket}$ is the time at which the ticket is purchased, and $Time_{remaining}$ is the time that remains from message acquisition until registration for tickets finishes. For instance, if a node that has interest degree 0.8 gets the advertisement at simulation time 5, and the registration ends at time 10, then the node will register for the event at time $Time_{ticket} = 5 + 5 * 0.2 = 6$. The formula has the semantics that someone who is very interested in an event will make sure to buy it quickly in the time remaining, while someone who is not quite that sure will take longer to decide. The choice of which nodes attend events is thus dependent both on their interest degree in it, and on when they learn of it.

The interest degree of each node is drawn from N(0.4,0.1), indicating that there are few very frequent event attendees (0.13% have an interest degree of more than 0.7), and most will attend events every now and then. It is independent of the interest, and any node in the network is assigned an interest degree.

#### Dissemination Behaviour

The last thing to define is the dissemination behaviour. For the purposes of my simulation, this is again a boolean. Its probability is proportional to the interest value, since the more interested and active someone is within a community, the more likely he is to stick to the common rules since he profits from it directly. This is a behaviour particularly difficult to model since there is little intuition or data that help here. I assume for simplicity that any node that has less than a defined value of interest degree is a bad disseminator. The precise value will vary throughout simulation runs depending on results obtained.

In general, I believe that such a multi-dimensional model of behaviour of a group of people only makes sense when seeing it as relative to a certain behavioural norm. Hence, an event behaviour which is considered highly inappropriate might be different at classical concerts than at rock concerts, but as long as we separate those two, and let behaviour refer to a particular type of event as I am doing by separating individual trust data by type of event, we can safely choose an intuitive distribution and claim that there can be an event with such a distribution among its attendees.

### 3.4.2 Mobility Model

The type of mobility model used for simulating realistic human movement is a statistical mobility trace, as outlined in 2.3.3. The lack of realism in synthetic traces outweighs in my view the advantages of implementational simplicity provided by these. Real mobility traces, given their small size and limited generality, are not an alternative either. A statistics based trace seems to best combine realistic human modelling, simplicity of usage and generality

I aim at a network consisting of 2000 nodes. This number is sufficiently large to give a meaningful simulation, and is also a conceivable size of such a network in a sizable city (for a population of say $\geq 200\,000$, we get about 1% to be oppnet users). I will compute a mobility pattern by first creating a set of nodes according to the statistical data and instructions given in 2.3.3. The relevant location data will be placed within the bounds of a real world city map. The paths taken from one point of the map to another will follow the street map of the given city, in order to simulate real movement more closely. A most desired feature would of course be to obtain data that concern the congestion of each street and pavement, or other flow related data. This would allow for even more realism, but it would complicate the simulation implementation beyond what can be done in the given time (particularly considering that none of the works consulted for this thesis was able to obtain or use such data).

I am postponing a more precise description of the mobility trace to the implementation chapter because this is a mostly technical task.

## 3.5 Security Considerations

There are some considerations concerning security that arise in the discussed context.

In the case of trust based security, there is always the risk of someone behaving well for a very long time interval, and doing something very detrimental once he is trusted enough. Fortunately, in the case of events, it seems difficult to see any incentive for that. Since events can in general be considered similar up to content, there is no reason to pick one in particular and behave significantly differently than at others (other than the capacity of any human to take unexpected action). Hence this otherwise quite serious issue is alleviated by the semantics of the underlying application.

An assumption inherent in the design of any trust based security framework in MANETs is that nodes are uniquely identifiable. Existing trust frameworks such as Ebay encounter the problem of malicious users simply being able to create a new account for the same trust framework when their trust value decreases, and being able to continue their business with a clean vest. This remains an unsolved problem by others, and I too will resort to assuming that devices are in fact

readily and uniquely identifiable. Such an assumption is also necessary to allow a trustee in my trust algorithm to verify if his predecessor has signed himself on the message correctly.

It would have to be guaranteed that a device that has never registered to join the network can't obtain a copy of the messages being exchanged. Now wireless message exchange can easily be overheard, so the only way to ensure that is to use some sort of encryption, or more probably define a special file format that can only be read by the trust application.

Almost certainly, this trust scheme would in practise raise privacy issues: After all, unknown devices access personal trust data upon each contact. I leave this issue unaddressed in this thesis. Possibly, some mechanism could be found that will allow to make sure that other people's trust data can't be stored on another device.

## 3.6 Conclusion

This chapter has introduces the detailed design decisions that affect the opportunistic network model used for later simulation. Specifically, the trust scheme has been described including the data stored at each node, the formula to combine these into a meaningful trust value, the dissemination control scheme and the communication protocol according to which message exchange takes place. Subsequently, the model of user behaviour has been presented which involves a model for event misbehaviour, of event interest, and of dissemination behaviour. A brief introduction to the mobility model was given, and security issues from a practical point of view have been discussed.

# Chapter 4

# Implementation

In this chapter, the technical implementation of the project is described. An overview is given of the general project structure, after which the components of this structure are described. Particular attention is paid to a thorough description of the mobility trace creator, which is the most challenging technical task that was undertaken during this thesis.

## 4.1 General Project Structure

My project is subdivided into two subprojects: The first one is concerned with the creation of mobility traces, the second with using them for message dissemination simulation. The implementation structure follows this division closely: It encompasses one application that is capable of creating mobility traces, an interface application that transforms the mobility traces into contact traces, and the dissemination simulation that uses the contact traces:

## 4.2 Mobility/Contact Traces

The downsides of real and synthetic mobility traces as described in chapter 2 have led to the decision of creating a new mobility trace in this thesis, which is based on statistical data about real human mobility. Below, I give a detailed description of the relevant design decisions and technical implementation thereof.

Conceptually, mobility traces are subdivided into two components. The first is the temporal component, describing the actions of each node and timings thereof. The second component is the spatial component, which maps these actions onto a chosen location.

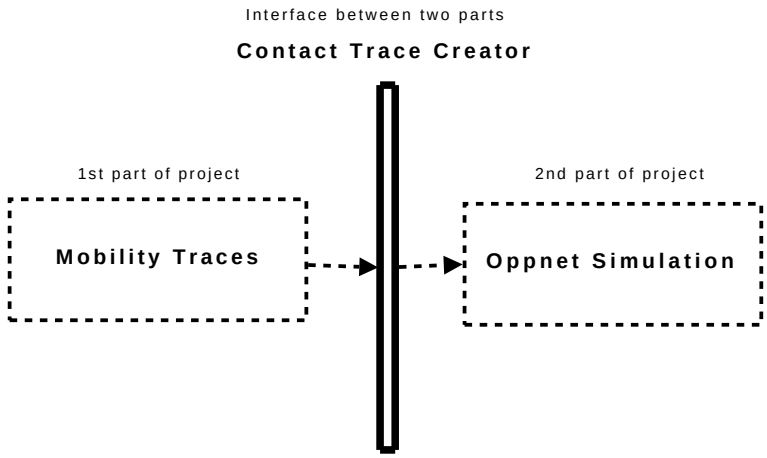


Figure 4.1: Project Structure

The software architecture underlying the implementation of the mobility trace creator is shown in 4.2:

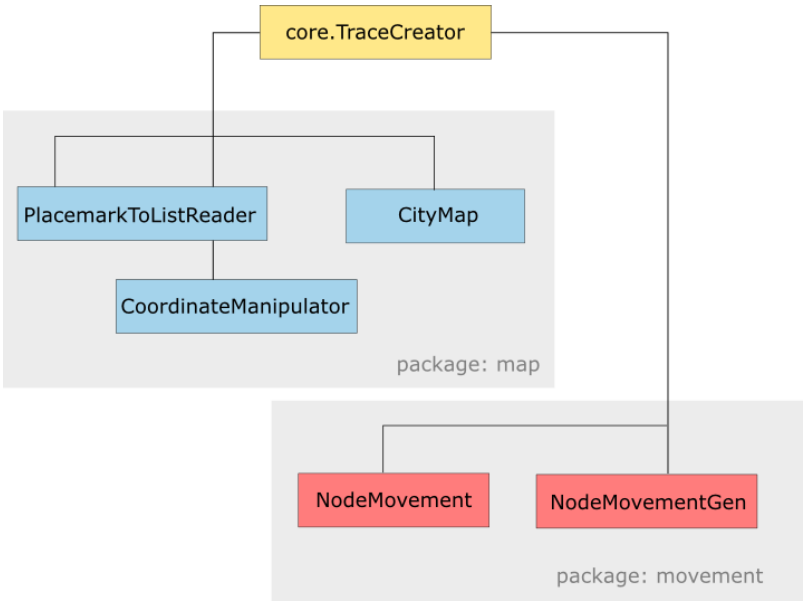


Figure 4.2: Software architecture of the mobility trace creator

The TraceCreator class is the main class that makes use of the functionality of the map and movement package classes to create a full mobility trace. To this end, it uses the class NodeMovementGen which stores statistical data about realistic human movement, and creates a temporal mobility trace for each node consisting of a sequence of activities that a node undertakes during each day, and the timing of these activities. The TraceCreator class uses the class CityMap

to map these activities onto locations on the map on which such activities take place. This results in a sequence of coordinates together with the time spent at each coordinate.

The map that is used for location is a real city map, therefore in order to link these activity location coordinates by a path that would realistically be taken from each location to the next, I use Google Maps. Google Maps has an inbuilt feature that allows to query the Google Maps server with a source- and destination coordinate, whereupon the server returns a file that contains the coordinates following the streets that a person would realistically take from one location to the other. The class PlacemarkToListReader takes as input such a downloaded path and incorporates it into the location oriented trace, to form a full mobility trace with coordinates that follow the street map of a real city in a realistic way.

Since the oppnet simulator takes as input contact traces, it is necessary to transform the mobility traces created by detecting when two nodes in it are within communication range. To achieve this I will use an existing tool, the ONE opportunistic network environment simulator detailed below, which has an inbuilt contact trace creator. However, the ONE takes as input mobility traces that describe mobility in Euclidean space, while the mobility traces that I create describe mobility on a real location on the surface of the earth, using geographic coordinates. Therefore, I need to transform the geographic coordinates into Euclidean space, the working of which is also described below. Overall, the workflow required for the creation of the mobility and contact traces is as follows:

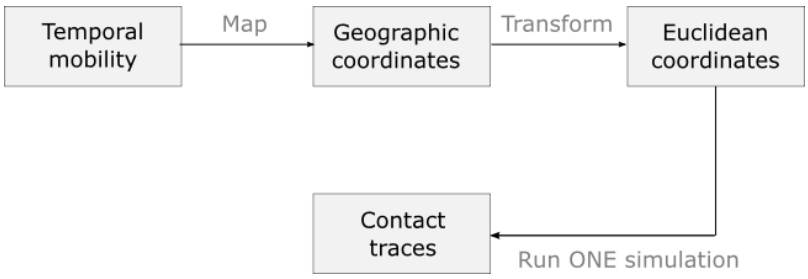


Figure 4.3: The transformations the mobility trace must undergo before being used for oppnet simulation

In figure 4.2, it is the package maps.geodesy that contains classes capable of translating the geographical coordinates of the mobility trace as created by Google Maps into Euclidean space.

The following subsections will describe the design decisions according to which these components have been implemented.

### 4.2.1 Temporal Component

To model the kinds and timings of activities of nodes, data from the US Bureau of Labour Statistics are used, as summarised in [17]. These data, according to the authors, have been collected from a 2003 study of 20,000 individuals (including some unemployed ones as well), asking them to record their working day routine. They are the most accurate and comprehensive set of data that I am aware of. The authors of this work have summarised the data available into distribution of

1. the time of arrival at work
2. the duration of the stay at work
3. the probability to take breaks during the working day
4. the start time of breaks
5. the number of activities performed during a break
6. probabilities of each possible activity performed during the break; the possible activities are eating out, shopping, going home, receiving professional service e.g. at the hairdresser's, exercising, going to relax e.g. in a green space, or dropping someone off somewhere (the latter representing any short trip for the benefit of someone else, such as bringing your grandmother her shopping)
7. the durations of each activity

They propose a simple algorithm that creates a mobility trace from these data: First, assign a random location to a node for home and for work. Then:

1. Determine the arrival time at work
2. Determine the duration of the stay at work
3. Determine if a break is taken
4. Determine the start time of the break
5. Determine the number of activities performed during a break
6. Determine which activities are performed during the break

7. Determine the duration of each activity

8. Determine the arrival time back at work and determine if a break is taken again (if yes, repeat steps 4-8).

Although of course the resulting mobility trace does not contain fine grained mobility patters such as walking around in one's office, it is well suited for the purposes of my thesis, since it is the more coarse daytime mobility that has the most influence on how messages spread in an entire city area.

This algorithm is followed very closely here, with a few exceptions: First, originally it is proposed to choose locations such as shops or places to relax at a distance from the current location that a person is likely to walk, according to a distribution computed in [3]. This would restrict the choice of these locations to places that are about 400m away from the current location. I have dropped this restriction as it doesn't seem to be realistic, firstly since it is common to travel using public or own transport in cities, and secondly because [3] has been written in 1975, and it is likely that urban mobility has changed since then.

Since the mobility trace is set in a real world map, the locations for each type of activity are chosen from the possible set of coordinates where such activities realistically would take place in the real location. More details are given in the next section since the choice of activity locations pertains to the spatial component of the mobility trace.

Another point to note about the data in [17] is that there are some inconsistencies in the distributions presented by the authors. When they give the distributions they have computed, they first present a graph of these distributions, but they also present a function which approximates the graphs. The graphs themselves are sensible, for instance the distributions of arrival times at work peak at 8am to 9am, and again at 10am to 11am, and decrease after that. Similarly, the other graphs are intuitive as well. However, the corresponding functions are mostly nonsensical, for instance they indicate probabilities that exceed one. The graphs are therefore used as a basis of the statistical data used in my trace, and I recompute the approximation functions for myself, this time making sure that all probabilities do integrate to one.

The third and last change that was made was to disallow my nodes to start activities when the time is close to midnight. The aim of this change was to ensure that the nodes reach their homes by midnight, so that each day can be treated independently by assuming that at the start of the day each node is at home, and simply concatenate many days, rather than having traces of different lengths for each day of each node that would then have to be integrated somehow. This simplifies the implementation, but does not change the traces much, since the average node takes only one to two breaks in the early afternoon, each of which involves one activity, therefore late evening activities are rare.

### 4.2.2 Spatial Component

Given the temporal trace created as in the last section, it is necessary to map the abstract locations such as "work" and "home" onto some two-dimensional space, to create an actual mobility trace. In order to stay similarly realistic in the choice of map as I try to be in the temporal component, the nodes are placed on a real map. The location is chosen to be the centre of the city of Warsaw, delimited by a white line in 4.4 - the precise delimiting coordinates given in decimal degree coordinates[1] are

**south:** 52.22128832297189 N
**north:** 52.25601038485794 N
**west:** 20.98400371188293 E
**east:** 21.03126925074066 E


Figure 4.4: The area of central Warsaw I have chosen for my mobility trace. The white lines show where the area is delimited.

The reasons for choosing this precise location are first that I know the city well, and am therefore able to determine where shops, parks and other points of interest are; this happen to be mostly well defined areas (except for eateries, which can be found everywhere). This knowledge allow me to place activity

[1]Typical, mutually equivalent formats for specifying geographical coordinates:
1.) DMS: Degrees:Minutes:Seconds (49°30'00"N, 123°30'00"W)
2.) DM: Degrees:Decimal Minutes (49°30.0', -123°30.0'), (49d30.0m,-123d30.0')
3.) DD: Decimal Degrees (49.5000°,-123.5000°), generally with 4-6 decimal numbers.
Examples from `http://en.wikipedia.org/wiki/Geographic_coordinate_system`

locations in a realistic way. Secondly, the location is delimited by a set of large streets, the importance of which will become clear in a moment.

To create mobility traces from the temporal trace, I first define a set of coordinates at which each activity may take place, which is fixed throughout the creation of all nodes. Depending on the activity, there are activities that may take place either at any location within the map's bounds (e.g. dropping someone off), or at a fixed set of predefined locations (going home clearly only has one possible location), or a mix of these with a certain probability for choosing either (eateries: each node has some favourites, but chooses probabilistically whether to go to a favourite location or to a random location). Then, the following steps are taken for each node:

1. Choose two random locations within the bounds of the city to represent a home and an office

2. For each possible activity, choose a small subset of locations that will be the nodes' 'favourite', i.e. most likely to visit locations (e.g. the favourite restaurants)

3. Draw the node's activities from the distributions given earlier. Whenever there is a location for an activity to choose, pick a location from the set of possible locations defined for that activity

The result of this algorithm is a sequence of location coordinates that follow the statistical behaviour of humans during the working day, and each entry in the trace is of the form

time - NodeID - x-coordinate - y-coordinate

Additionally to moving nodes representing humans, I allow to specify as an argument to the mobility trace creator a number of additional static nodes. There are several works concerning opportunistic networks - for instance [12] - propose to enhance pure opportunistic networks with a small set of fixed-location "information sprinklers", i.e. a wired backbone that receives information from passing oppnet nodes and make it available instantaneously at all other backbone nodes, through Internet connectivity. This is a useful feature to be able to create, particularly since it is likely that in first practical applications, some sort of base stations will be utilised (as in it happens in [2] already), rather than trying to implement a pure oppnet.

The last transformation that needs to be done to turn this into a mobility trace is due to the fact that a path from one point to the next in a node's "schedule" is not straight but angular, since it follows the street map of a city. To this end, feature of Google Maps is used that allows to input into the address bar a

source- and destination coordinate, and that outputs file in KML format[2] which contains a path from the source to the destination that follows the given street map. It is here that the delimitation of the chosen city area by a set of major streets becomes important. Consider that it is not possible anticipate which of several alternative paths Google Maps will return when being given a source and destination point. In particular, there is a danger than Google Maps will lead the nodes along a path outside of the predefined city area, which will cause the simulations to abort. If however the area is delimited by major streets, then any path between two points will never leave these boundaries, since it will eventually have to return to these major streets to reach the point which is within the area limits. This would be a detour, and Google Maps does not return paths that are longer than necessary in terms of path length. Fig. 4.5 and 4.6 should clarify:

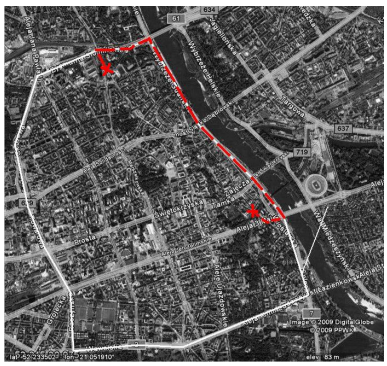

Figure 4.5: A likely path of a node returned by Google Maps given source and destination. It leads along the major streets, the node stays within the area limits.

It is important to note that the coordinates that are used up to this point are coordinates on the surface of the earth. To be precise, I am using the Geographic Coordinate System, with the datum WSG84. "Geographic Coordinate System" denotes the coordinate system of latitude/longitude pairs that is commonly known, and the "datum WSG84" refers to the definition of the center of the earth and its shape, which all together define where exactly a coordinate is on the surface of the earth. I am using WSG84[3] since this is the datum used by Google Maps, as an aside it is also used in GPS.

When inputting path data into Google Maps, it is possible to specify whether the path returned should be suitable for driving by car or for walking by a human. This distinction is not made here, since the nodes are moving right in the centre

[2]KML is the file format used by Google Earth and Google Maps, and is defined as a standard of the Open Geospatial Consortium, Inc. on `http://www.opengeospatial.org/standards/kml/`

[3]`http://www.ordnancesurvey.co.uk/oswebsite/gps/information/coordinatesystemsinfo/guidecontents/guide4.html`

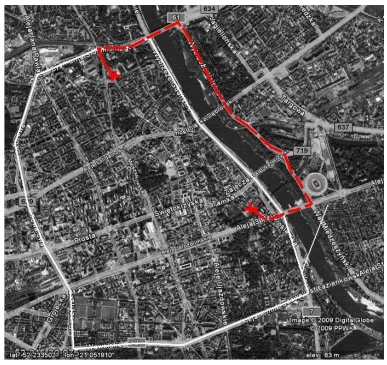

Figure 4.6: An unlikely path: The node chooses a similar street that is further away to travel. It does not stay within the area limits.

of the city, where there are very few streets or paths where cars may not go, and so these paths are equivalent. Note that this feature only affects the coordinates along the path of the nodes, and not the speed with which a node moves.

As a last step, a way of assigning the speed at which the nodes move needs to be defined. Here, I am assuming two distinct speeds, one for walking, which is set at random between 3km/h and 6 km/h, and another speed of nodes that use public or private motorised transportation, which varies between 30km/h and 45km/h. I choose these values as an average, considering than in the city centre, cars may have outbursts of driving at 70km/h, but also stop frequently because of bad traffic. Due to the underlying implementation, the speed of the cars varies up to about +/- 15%, so that this average is not unrealistically static.

### 4.2.3 Advantages and Shortcomings of the Mobility Trace

In short, the advantage of the mobility trace as described above is that it can be used for a much wider range of purposes than real traces because the statistics it is derived from is collected in a way that is independent of the geographical environment. The resulting trace is hence independent of the geographical environment as well. Additionally, it allows to create an arbitrary number of nodes while retaining the statistical properties of the original trace, in contrast to real mobility traces which are restricted to a fixed, usually small number of nodes. Since the statistical data are essentially real mobility data, statistical traces are in fact equally realistic as traces that reuse real data[4]. The independence of location allows the users of the trace to choose the area that is most suitable to

[4]At first sight, it might seem that real mobility traces are much more fine grained than statistical data, i.e. that real mobility traces might for instance contain the exact paths of a human including how he moves around within his office. This is not the reality at the moment; real mobility traces concerning humans are often collected by letting the human's device log the wireless sighting of cell towers distributed in the area (for instance see [8]). Clearly this mobility

their application, and to use Google Maps to create a realistic travel path. Note that Google Maps is aware of one-way-streets and suchlike features of the streets, so it is by far more realistic than for instance Dijkstra's shortest path algorithm, used for instance in the mobility trace in [9].

The shortcomings of the presented mobility trace are first of all that the statistical data used for the temporal component are incomplete, in that they don't take into account the time spent before/after the working day has begun/ended. Furthermore, the spatial model does not take into account any obstacles on the way of the nodes such as traffic lights or jams, so that all nodes move around independently of each other, clearly an unrealistic assumption. It is due to the fact that such, traffic data are very difficult to obtain from the respective traffic authorities , and secondly because they would be infeasible to implement in the given time.

### 4.2.4 Contact traces

The trace created so far is a pure, location-oriented mobility trace. The final aim however is to use this trace to determine the co-location patterns of the nodes, i.e. to create a contact trace, since we are interested in message transfer opportunities. To this end, a simulator devised at the Helsinki University of Technology is used, which takes as input a mobility trace, and outputs a contact trace. It is the ONE (Opportunistic Network Environment) simulator, it webpage can be found on [13]. The parameter that can be set with respect to the network characteristics, and that is relevant for my application, is the connectivity range of the nodes. The range is set to roughly correspond to Bluetooth standards[5], to model this ubiquitous and lightweight technology that humans are likely to have in their mobile devices. Hence, the connectivity range is set to 10m. Since only contacts between nodes are relevant, there is no need to set parameters concerning messaging such as transfer speed. However, I do make some assumptions about these in the later dissemination simulation, namely that neither restricts the connections, i.e. that all messages that need to be transferred can be both fully transmitted and stored at each contact. Modern mobile devices store music, video and games on them, so they are certainly capable of storing ordinary text/image files that the event ads are likely to be. Hence, these assumptions are realistic, while on the other hand simplifying my dissemination simulations by allowing me to leave out network parameters from the simulation parameters.

The transformation that needs to be performed to use the mobility traces in the ONE is to convert the geospatial coordinates into euclidean coordinates. To this end, I define a 'zero' point in the bottom left corner of the map of

trace is equally coarse as a statistical trace, it only tells us "at time x, node n has been within the range of celltowers u,w, and v".

[5] see [33]

Warsaw together with an x-axis and a y-axis that go through this point and are parallel/perpendicular to the equator. Then for each point in the trace, I compute the spherical distance of each x/y coordinate to the corresponding axis, and use these distances as the two dimensional coordinates, see fig. 4.7.

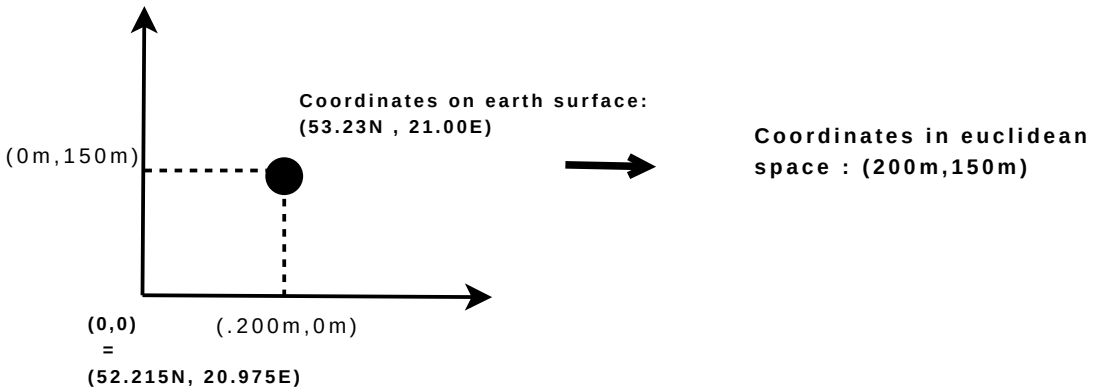


Figure 4.7: Project Structure

An alternative to the simple, distance based transformation used here there are many other coordinate projections such as the well-known Mercator projection that Google Maps uses to represent flat images at low altitudes. But for my purposes, a projection is necessary that preserves the distance between the nodes, otherwise the connectivity range of the nodes is not preserved. The distance-based transformation is suited to the application (in contrast to e.g. the Mercator projection, which does not preserve distances), so there is no need to implement any more complex projection.

### 4.2.5 Shortcomings of the Contact Trace

The contact traces I create follow existing wireless communication specifications, but they do not take into account physical obstacles such as buildings or trees that could interfere with the contact, therefore representing the upper bound of possible contacts.

On the other hand, contact opportunities arising from evening activities outside of working hours and from attending the same event, which might contribute to message spread, are not recorded by the underlying mobility trace. All in all, the distribution of contact opportunities during the day is shifted towards contacts happening during the working day hours. It would have to be determined in how far this affects the contact trace, e.g. it is quite possible that the number and kind of contacts in reality would actually be the same, since many people attending the same event all exchange messages, so the contact opportunities not

given during the working day due to physical obstacle are made up for at such gatherings and during other evening activities.

## 4.3 Oppnet Simulator

The dissemination oppnet simulator has been implemented closely according to the model design described in chapter 3. A simplification that need to be mentioned has however been made.

According to the communication protocol detailed in sec. 3.3.2, the nodes have to verify the integrity of each others' trust data to make sure these have not been altered. I am not implementing this in the dissemination simulator. Semantically, this excludes nodes from the network when they are reported, and excluded nodes don't contribute anything to the simulation given that the total number of remaining nodes is large enough.

## 4.4 Technical Tools

I acknowledge the use of several external packages to the Java programming language that I have used for this implementation.

For any computations involving probabilistic components such as random number generators or sampling from standard probability distributions, I rely on a statistics package created by Michael Flanagan at UCL. The package is comprehensive enough for me to use it both for the creation of the mobility trace, and later on for the message dissemination simulation as well. It is available at [10].

For computations involving geospatial coordinates that I am using to create the mobility traces, I chose the package org.gavaghan.geodesy available at [11]. It uses the so called Vincenty Formula to perform the complex computations required to find distances and related values in geospatial coordinates. The Vincenty Formula has an error term of at most 0.5mm[6], which is currently the most accurate formula to the best of my knowledge.

The graphs shown the next chapter are created using gnuplot, the homepage of this tool is `http://www.gnuplot.info/`.

## 4.5 Conclusions

This chapter has described the overall project structure in terms of the technical implementation. It has given all important technical design decisions that have influenced the final implementation, with particular attention to the mobility trace creation. It has been explained which feature of the mobility traces make

[6] See [35], on page 91

them superior to other existing traces, and how care has been taken to ensure a high degree of realism in every aspect.

# Chapter 5

# Evaluation

## 5.1 Overview

In this chapter, the simulations of message dissemination and trust evolution that I have run using the previously created contact traces are described. During these runs, I will collect information about

1. how many nodes in total are reached by the advertisement - to determine the effectiveness of the oppnet as a dissemination network
2. the fraction of interested nodes reached - to determine with which simulation settings this fraction is maximal, since this ensures revenue for the organiser. Additionally, this fraction shows how many nodes are excluded from the network on account of a low trust value in the course of time.
3. the damage done by malicious nodes at events - the value to be minimised
4. the average misbehaviour probability of the reached nodes - as a reference value to determine how representative the event attendees are in the total reached population, i.e. if the damage done at the event is an expected value, or larger or smaller than expected by chance
5. the number of trust data available for each node - represents the accuracy of the trust value computed
6. the number of bad disseminators in the reached population - allows to determine how large the percentage of bad disseminators can maximally be for the network trust scheme to work
7. how the above outputs change over time depending on the set up, and what the ideal simulation time is for each set up (i.e. for how long to

keep a message in the network in order to reach the maximum number of interested nodes, while keeping the reached malicious nodes at a minimum)

The sequence of simulations planned to determine the network characteristics under the effect of trust and user behaviour is as follows:

1. Determine the sensitivity of the network with respect to dissemination speed to the speed of the nodes in the underlying mobility trace, and to the number of initial messages in the network. These data are necessary to obtain in order to assess how generally valid the later dissemination simulation results will be, and in consequence if these results need to be obtained for several network settings.

2. No requirement of a trust computation: The data obtained during these simulations will show how the system would behave if left on its own, without the application of trust to increase security. Nodes are thus allowed to disseminate messages to any node in the vicinity, without testing other nodes' trust values.

3. Requirement of a trust computation, all nodes cooperate: This run represents the ideal behaviour of the system, and a benchmark of how good the trust scheme can maximally be when all nodes apply it when compared against the no-trust run.

4. Requirement of a trust computation, differentiated dissemination behaviour: A comparison of this run's results against the ones with the all-good dissemination run will show in how far bad disseminators affect the network, and how large the fraction of bad disseminators needs to be in order to be noticeable.

5. Dissemination control: These runs will show the effectiveness of the dissemination control scheme implemented, as well as any conditions that may restrict it.

Each of these simulation runs is explained in a separate section below, after a section that is dedicated to explaining the general settings of the network that apply throughout all simulations. The findings are then summarised at the end of the chapter in section 5.8.

## 5.2 General Settings

There are in total three contact traces that are used for the simulations. The mobility traces that underlie these contact traces consist of 1800 nodes each. Each of the traces covers a different speed interval for driving speed, walking speed is

not taken into account because the walking speed settings do not differ greatly from mobility trace to mobility trace, see 4.2.2. The speed intervals are for driving speed considered are [30km/h,35km/h], [36km/h,38km/h] and [40km/h,45km/h]. This subdivision of the overall interval of [30km/h,45km/h]used to create the mobility trace (see sec. 4.2.2) allows to test whether node speed affects message dissemination.

I am assuming one event organiser who organises a sequence of events of the same type. There is no need for more organisers since several organisers running the same type of event are equivalent to one organiser running this type of event more frequently. A node is interested in the event with probability 0.1, i.e. about 180 nodes are interested in the event. The number of nodes that may attend any event is by default 70. I am simulating 28 consecutive days from 14 days of contact traces, by first stepping through all 14 day traces in order, and then reusing them in a random order that is fixed for all runs and all traces. Since the contact traces for separate days are independent of each other, this creates a uniformly distributed random ordering of the days. The introduction of a random ordering ensures a greater degree of realism, since human mobility is not completely regular and repetitive.

An initial observation of simulation results has led to the conclusion that in general, most nodes in the network are reached by the message within one day if no constraints are applied to the dissemination (see 5.3). The data collected for each simulation are therefore collected for each day separately, possibly involving a change of parameters such as the nodes' trust data from day to day. In reality, events may of course take place more rarely, in which case the event organiser would have to implement an expiry time after which the message is not spread any more to prevent it from cluttering the network.

## 5.3 Benchmarking Simulations

Before focusing on the trust mechanism and user behaviour, I have run two benchmarking simulations. First, I have determined whether the speed of the nodes influences the contact traces, i.e. if for instance higher speeds cause the contact trace creator ONE to not record contacts that it would have otherwise recorded. According to the parametrisation given to the ONE contact trace creator, contacts are sought for every 0.5 seconds. The non-continuous contact detection corresponds to reality in that mobile devices scan periodically for contact opportunities. Note that the periodic scanning is likely to causes some possible contacts to be overlooked. Given this periodical scanning, even at the highest speed of 45km/h (when a node can travel 6.25m in 0.5 seconds) this means there should not be significantly less recorded contacts than at low speed, given a connectivity range radius of 10m. And indeed, it takes exactly one day (86400 seconds) for

each node in the population to receive the message, if it is freely disseminated without regard to trust. This result applies irrespective of the speed, as fig. 5.2 and 5.1 show for the extreme cases.

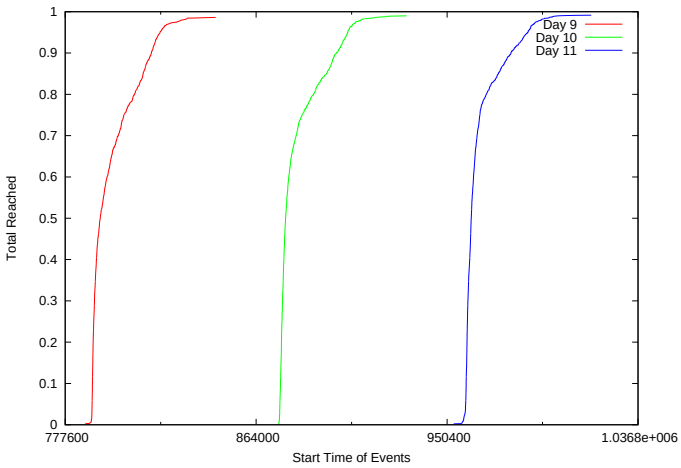


Figure 5.1: Reached nodes over the course of time given a node speed between 40km/h and 45km/h

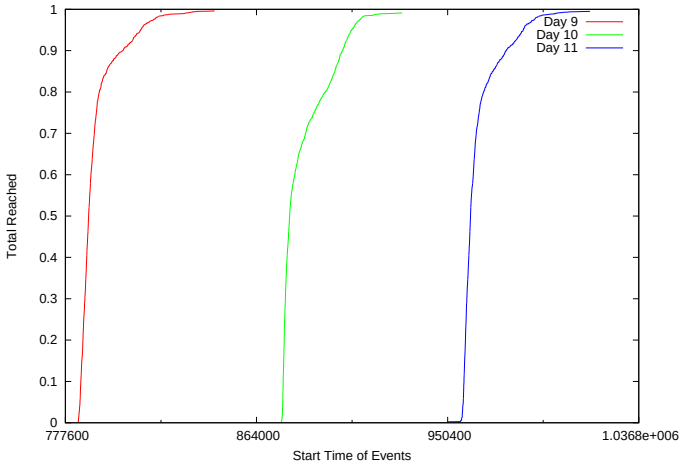


Figure 5.2: Reached nodes over the course of time given a node speed between 30km/h and 35km/h

On both graphs, we can first see an initial low activity time, which corresponds to the early hours of the day when the nodes are "asleep" according to the statistical movement data. At a time corresponding to 5am on each day, the saturation of the network by the message increases very quickly up to a fraction of about 97%, at which time there is a visible flattening out of the curve. I will from now on use the three disjoint-speed traces interchangeably, and include the results using one of them in this chapter.

The runs above have been made using only two nodes as initial disseminators, i.e. the initial number of messages in the network is two. It is also important to determine in how far the initial number of messages influences message dissemination. When changed to 40 initial messages, the time to saturate the population decreases visibly, but not significantly, as shown in fig. 5.3.

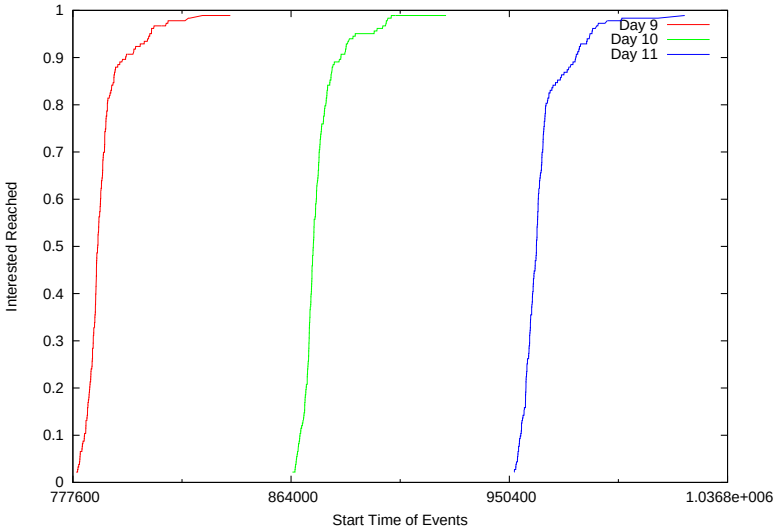


Figure 5.3: Node speed between 36km/h and 38km/h - 40 initial disseminators, in contrast to two in each of the previous figures

We can see that the slope of the curve is steeper than in the previous curves This, together with the flattening out of the curves towards the end of the day, shows that the network must be subject to some degree of clustering and even partitioning (as networks involving humans are likely to be), the latter since the saturation is not complete for some days - a very small percentage of the nodes is never reached at all, shown by a gap at the top of the curve. This again means that changing the number of initial messages in the network will change the dissemination speed by a small factor, but as long as I don't increase the number

of initial disseminators further should not affect the simulation results. In the later simulations, there are always 10 initial messages in the network, and these disseminators are always chosen to be the same nodes to ensure comparability.

Notice that there is a small gap between the x-axis and the lowest point of the curve. This is due to there being a number of initial disseminators who have the message before dissemination starts, so that the number of reached nodes is never zero.

## 5.4 Dissemination without Trust Based Security

The runs described in this section assume that any node is allowed to receive the message that is being disseminated, irrespective of its trust value or other restricting factors. Note that for this no-trust setting, all days show equivalent graphs, since no nodes are excluded from receiving the message at any time. I will now describe all data that I collect for any simulation setting and give example graphs for them using the no-trust run. This listing serves as an introduction to the network characteristics pertaining to trust, and as a reference that gives a basis for comparison for later simulations as well. For most of the data, the curves for all 28 days are subdivided among several graphs to improve readability. I show either the days that are most interesting, e.g. because they show a particularly visible change or because similar data have been shown on this day as well, or if it doesn't make a difference, I choose a day at random,

Curves describing the fraction of nodes reached by the message out of the total number of nodes are referred to in the previous section.

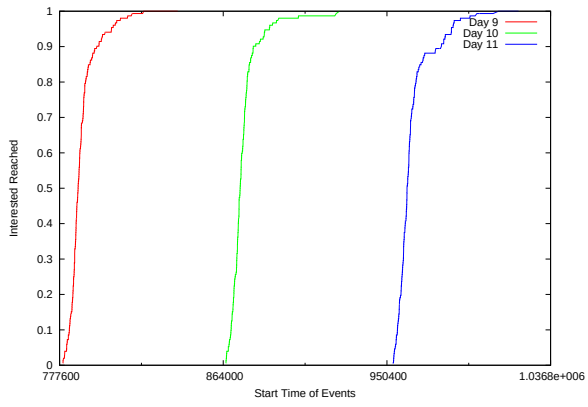


Figure 5.4: Reached interested nodes in a network with no trust computation requirement

Another important data set measures the number of reached *interested* nodes. The corresponding curve is given in fig. 5.4. This curve follows the exact same curve as those given in the previous section for the no-trust run, since interested nodes are distributed uniformly in the population, and the message is disseminated to any node in the vicinity irrespective of its interest state.

To get an intuition of how many nodes in the network are undesired at events because of their bad misbehaviour, consider that a node's mean event behaviour is drawn from N(0.1,0.1) (see section 3.4.1). This means that nodes that misbehave on average - i.e. whose average event behaviour is above 0.2 - make up 19% of the network. The trust scheme should hence cause a drop in interested reached nodes by at least 0.2 in order to be effective, ideally more to weed out "border case" nodes.

I also measure the number of trust data that are available to make trust decisions for the interested nodes (uninterested nodes won't come to events, so their data are not relevant), and record their distribution.

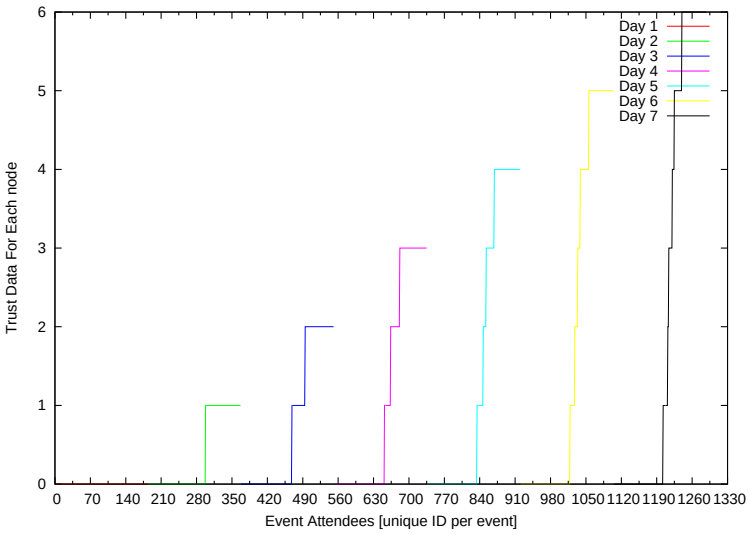


Figure 5.5: Distribution of trust data among the interested nodes, no trust computation requirement

The graph in fig. 5.5 represents the occurrence of each number of trust data any event attendee has, sorted by increasing number of trust data (e.g. a node that has attended five event so far has five trust data points - if there are many such nodes, the vertical line at y = 5 will be wide, otherwise narrow). The x-axis of this graph represents a unique ID given to each node at each event. This

allows us to see first of all how many nodes attend an event in total from the span of each separate curve, and how many of these attendees had exactly how many trust data. For the no-trust run, graph 5.5 shows that for the first event, no node has trust data yet, so the total number of trust data is 0. After each event, the previous attendees have one more data point than they had before. Hence for day 1, there are some nodes with one trust data point, and some with none, and for day 2, there are nodes with two, one and zero data points. For the future discussion, it is interesting to note that there is a relatively high number of nodes at each event that have the maximum possible number of trust data points, which is as many as there have been events so far. These are the nodes that are both close in the network to the initial disseminators and have a high interest degree. They are admitted to the event whatever their previous behaviour, and their high interest causes them to buy tickets early, so they always attend the events and accumulate a large set of trust data.

This corresponds to fig. 5.6, which shows that the number of trust data per event attendee of each event, shown for a larger number of events, grows linearly.

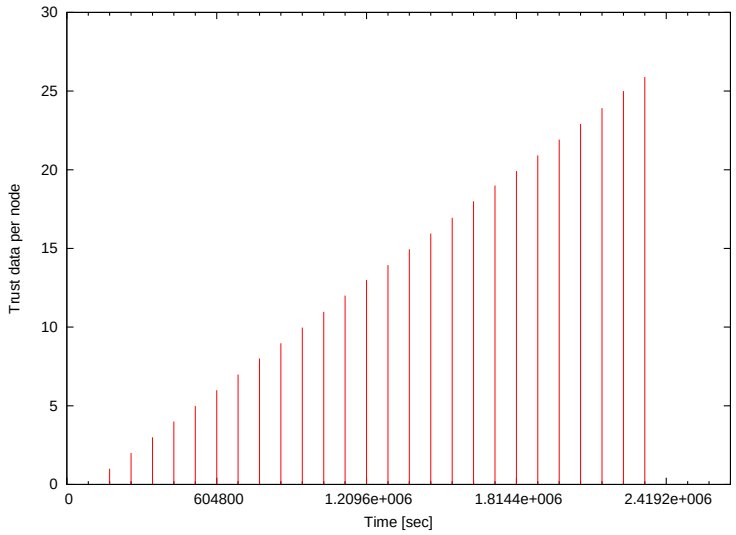


Figure 5.6: Accumulated trust data per attendee for several consecutive events, no trust computation requirement

Fig. 5.7 shows the behaviour that event attendees have on average, which corresponds to the damage done at the event. The ultimate goal is to lower this value through application of the trust scheme, ideally to less than 0.2, which is the smallest behaviour value considered to be slight misbehaviour. This can in

fact be achieved even in the no-trust scenario, as another example run with the same parameters shows: There, the average interested node misbehaviour is very close to 0.2 as averaged over all events, i.e. the average node is better behaved than those that I have shown. This is due to the average misbehaviour of *all* interested nodes, also measured, being by chance lower than in the first case, which clearly causes the average event attendee to be better behaved as well. However, both these graphs show a large variation from event to event, while we would prefer the damage to be more predictable. Additionally, the damage done in both cases is larger/smaller than 0.2 due to chance, being proportional to the randomly chosen behaviour distribution the nodes, while we are interested in achieving a steady occurrence of values smaller than 0.2, irrespective of the initial behaviour distribution. This captures our intuition that different sets of humans will behave differently on different days, and thus introduce variation into the event damage. If any person regardless of its personality is admitted to any event on a first-come first-served basis, we expect that there will be large variations in event damage, depending on which persons happened to buy tickets first. The event organiser however wants this behaviour to be predictable and stable, and as low as possible.

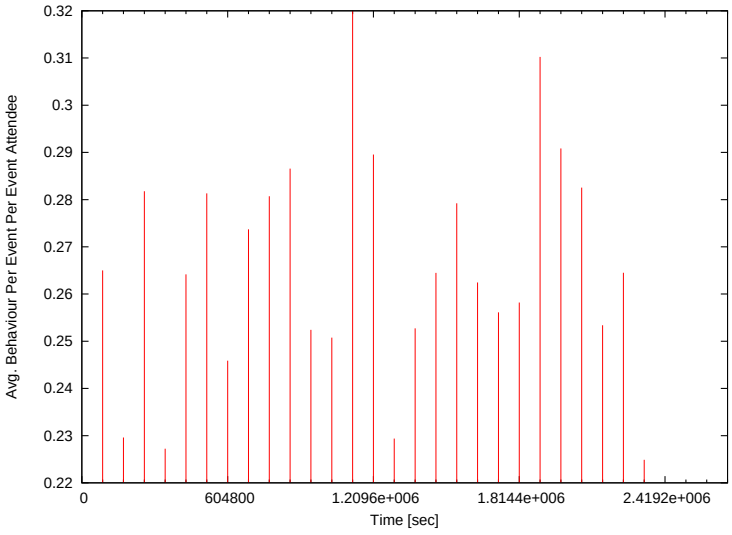


Figure 5.7: Average behaviour of all attendees at all events in sequence, no trust computation requirement

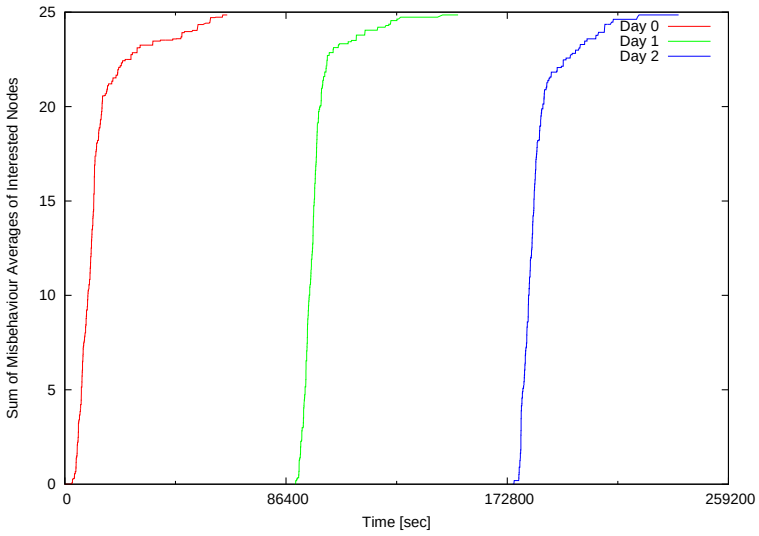


Figure 5.8: Accumulated average misbehaviour of interested nodes, no trust computation requirement

The average misbehaviour sum (equivalent to the mean of average misbehaviours) of all reached interested nodes, as opposed to event attendees, is shown in fig. 5.8. The curve intuitively follows the reached-nodes curve for this run, peaking at about 25, since these data are accumulated during the dissemination. This gives an average misbehaviour mean of $\frac{25}{180} \approx 0.138$ per interested node, and corresponds to the distribution of misbehaviour averages being N(0.1,0.1). For different runs, this accumulated mean of misbehaviours differs. Usually it stays within the interval [23,27], and in the extreme cases may reach 18 at least and 30 at most. I use only those runs for which it is between 23 and 27 to show graphs in this chapter, to ensure comparability of absolute values. This value is important to take note of before comparing any other run results, since it is directly proportional to event damage: A low damage achieved given a high misbehaviour sum of all reached interested nodes is more meaningful than low damage given than the nodes on average have low misbehaviour anyway.

These last data relate directly to the objective of this thesis: We want to make sure that the average misbehaviour of potential event attendees as well as the corresponding actual event damage is as small as possible, while the fraction of interested reached nodes remains high, i.e. we want to reach the largest possible number of "nice" nodes, while avoiding the ones with high misbehaviour probability.

## 5.5 Trust with Cooperative Dissemination Behaviour

This section describes the set of runs in which computational trust is applied to keep the message among trusted nodes, with all nodes in the network cooperating by disseminating the message according to trust values. The trust computation is done as described in sec. 3.3. By default, I set the utility loss of slight misbehaviour to 0.4, of moderate misbehaviour to 0.7, and of severe misbehaviour to 0.9. These settings have been determined by trying out several settings, and chosen to make sure that the risk factor of the trust formula has appropriately high weight. Additionally, I allow for a warm up time of three events for each node to be able to establish a minimally representative trust record. I.e., for the first three events, all nodes are considered trusted by being assigned the initial trust value, and only after that relevant data are used to infer trust. A consequence of this is that the first three-day graphs of any simulation results show curves that are equal to those of the no-trust run.

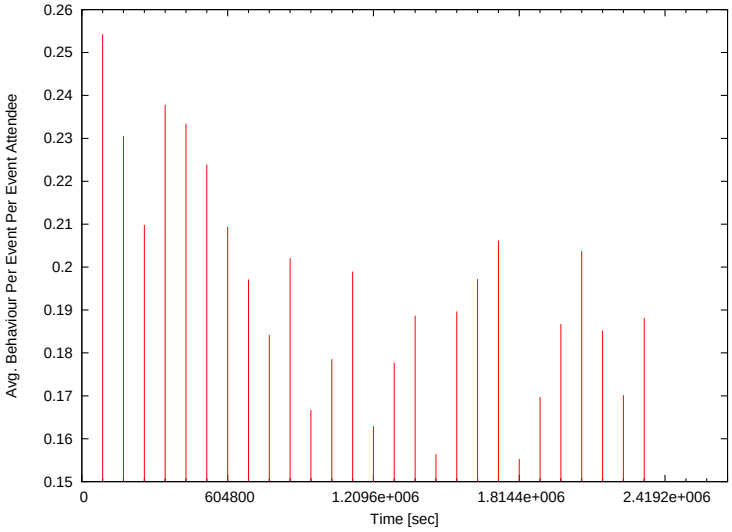


Figure 5.9: Behaviour of event attendees improves significantly when using trust based security and a trust threshold of 0.5

For a trust threshold of 0.5, there is immediate improvement in terms of damage in comparison to the no-trust run: In fig. 5.9, we can see a considerable decrease of damage to a level well below 0.2 with time. There still are variations from event to event, but at no event from the eighth event onwards does the damage exceed the critical value of 0.2. We can also see that the number of interested reached nodes decreases down to 70% after 27 events (see fig. 5.10),

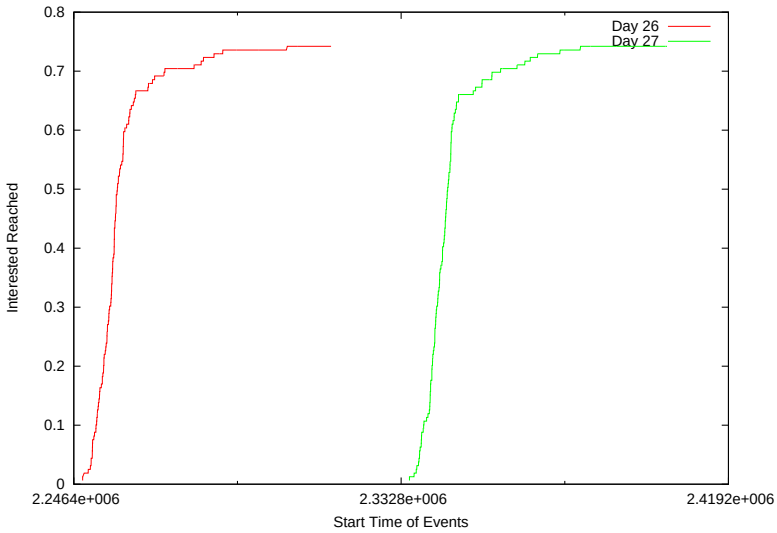


Figure 5.10: Fraction of reached interested nodes after 27 days with a trust threshold of 0.5

since some of them are excluded from getting the message on the basis of their bad event behaviour. The decrease of this fraction is gradual over the days, which comes from the event attendees gradually accumulating trust data, during which process more and more of them reach a trust value too low to be trusted again. Note that the fraction of total reached nodes sees less of a decrease: Since only interested nodes can go to events and be excluded from the network, the difference in interested reached nodes will naturally be more significant than that of all reached nodes.

As expected, increasing the trust threshold to 0.6 again improves the damage at events, at the expense of interested reached nodes, which drop to about 65% after the 27th event. Similarly, a threshold of 0.7 means a decrease of reached interested nodes to approximately 52%, while the event damage drops even further. In all cases, the decreasing takes place gradually, from day to day, so that possibly these values could see a slight amount of further change if simulated for another few days.

In all these cases, the drop in interested reached nodes is more than 0.2, which was set out as a lower limit of effectiveness in sec. 5.4. This shows that the trust scheme devised for this network does indeed provide an enhancement in terms of security, under the assumption that all nodes cooperate by disseminating according to it. It can be chosen as needed, depending on the maximal misbehaviour desired without obvious restrictions.

With respect to trust data, I observe that an increase in the trust threshold means that there are less nodes with the maximal number of trust data (see 5.11) in comparison to the no-trust run shown in 5.5, to see this compare the widths of the top horizontal bars in the curves between the two figures. This comes from some of the more interested nodes, which go to events frequently and accumulate trust data, being excluded on account of previous misbehaviour. They are then replaced by new attendees with fewer trust data that didn't have the opportunity to misbehave yet or those that have a smaller interest, and thus didn't manage to buy a ticket previously. The effect is that the distribution of the number of trust data becomes spread out more evenly. For the dissemination network, this means that many nodes in the network get the chance of attending some of the events, without there being a small group of nodes that always gains access first.

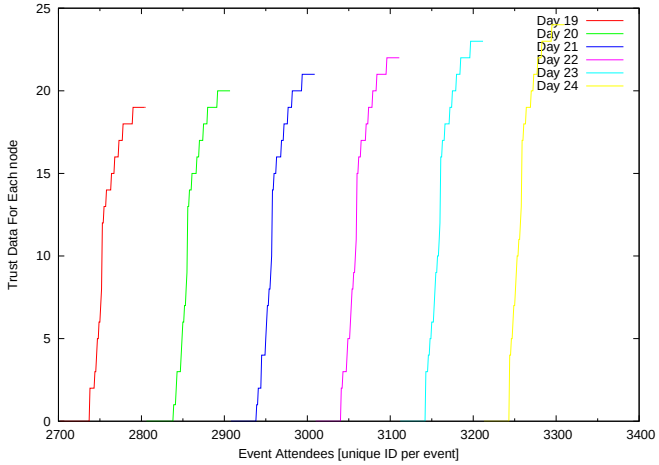


Figure 5.11: In comparison to the no-trust run, a threshold of 0.7 with cooperation causes the number of nodes with the maximal number of trust data to decrease

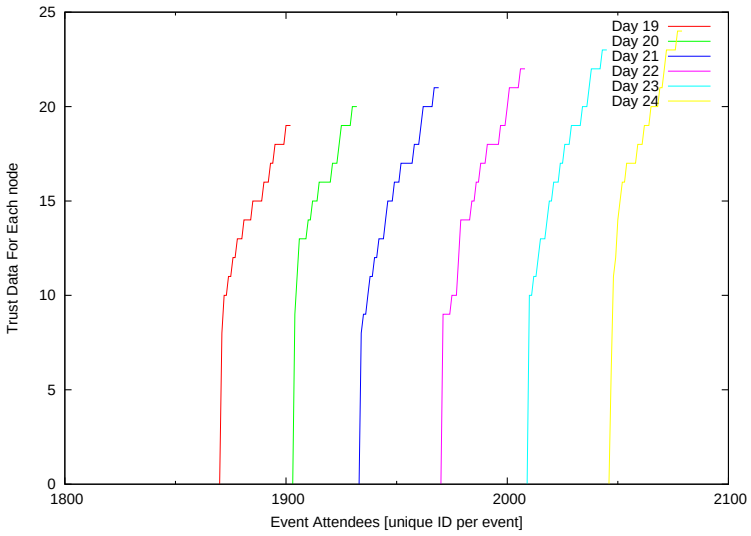


Figure 5.12: Visible drop of event attendees to about 35 (from 70) when increasing the threshold by 1.01 per event

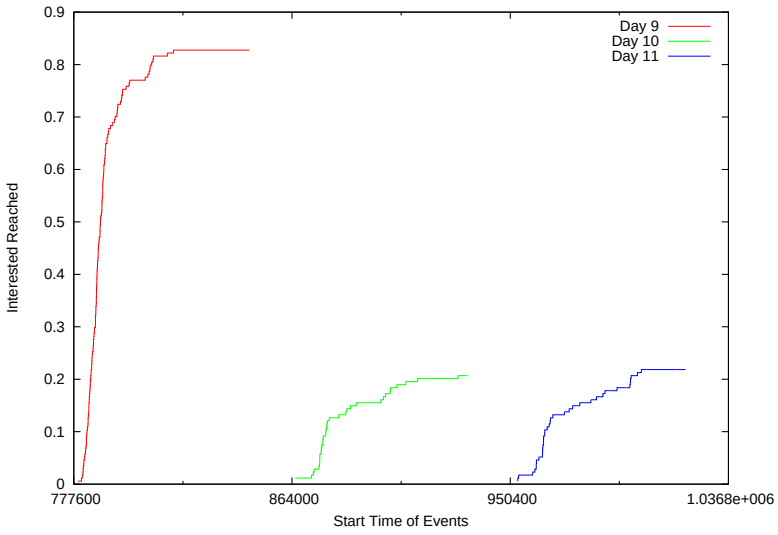


Figure 5.13: Sudden break down of reached interested nodes when subjecting the trust threshold to an increase of 1.01 per event

Consider that the trust I am computing is proportional to the ratio of good to bad event behaviours of a particular node. If out of five events, a node has misbehaved at one, giving a ratio of bad to total behaviour of 0.2, an organiser might well agree to still admit it to another event. If however a node misbehaves at ten out of 50 events, which gives the same ratio as before, we would expect an organiser to not allow this node at his events anymore. Hence, we expect that the trust value should increase for a well-behaved node with time, so that bad behaviour can be treated as a one-off occurrence. I have tested how my network would develop under this assumption, which I have implemented by continuously increasing the trust threshold with time, up to some limit.

In the first run, the threshold is multiplied by 1.01 after each event. After 27 events, it thus reaches about 0.65 when starting at 0.5. The effect of this increase is good with respect to the event damage, which decreases to an average of slightly below 0.18. However, when inspecting the distribution of the number of trust data points per attendee, we see that the number of event attendees has decreased significantly, see fig. 5.12, down to about 35 attendees per event. This can be seen when On the one hand this means that these attendees are very well behaved, on the other that the organiser loses revenue - it would be better to maximise event attendance while minimising damage. Additionally, we can see in fig. 5.13 that the number of reached interested nodes drops suddenly to about 20% on day 9, and later it remains at this level until day 27. Given the results for runs where the threshold is stable, this is clearly not an alternative trust scheme setting. Repeating the above runs while increasing the threshold by 1.04 at every run, even if capping the growth at 0.6, is strongly detrimental to the network, in that event attendance, the fraction of interested reached nodes and all other data drop to zero immediately after the warm up time. Clearly, the threshold increased too quickly to allow the nodes to accumulate positive trust, and all nodes were excluded from the network immediately, without getting a chance to rectify the trust judgement. This shows that the trust scheme should not be chosen to be too strict, since malicious nodes can always be excluded, but non-malicious nodes cannot be brought back to the network once injustly punished. For an event organiser, this means that it pays off to be optimistic with respect to the network users, rather than being overly careful.

In another run, in which I set the threshold increment to a small 1.005 - the threshold reaches at most 0.58 - the drop in all data is less drastic than in both of the previous examples, however it is still great. The overall damage drops to about 0.16, a very good result, but similarly to above event attendance is too small, see fig. 5.14. We can see that there no attendees with few trust data points from a certain point in time onwards. This corresponds to the intuition that when keeping the trust threshold at one level, we allow new, well-behaved event attendees who never attended events before and thus couldn't accumulate positive

trust, to attend the events. In contrast, gradually increasing the threshold weeds out these rare, well behaved event attendees, but also nodes that behave badly every now and then, their trust value floating just above the threshold all the time. The only attendees that can come to events subject to a gradual trust increase are thus those that constantly and frequently behave well at events. Consider that only the trust threshold has been increase, and not the initial trust, therefore any node that isn't able to accumulate enough trust data quickly will be excluded from the network by being assigned a trust value lower than the threshold from some time onwards.

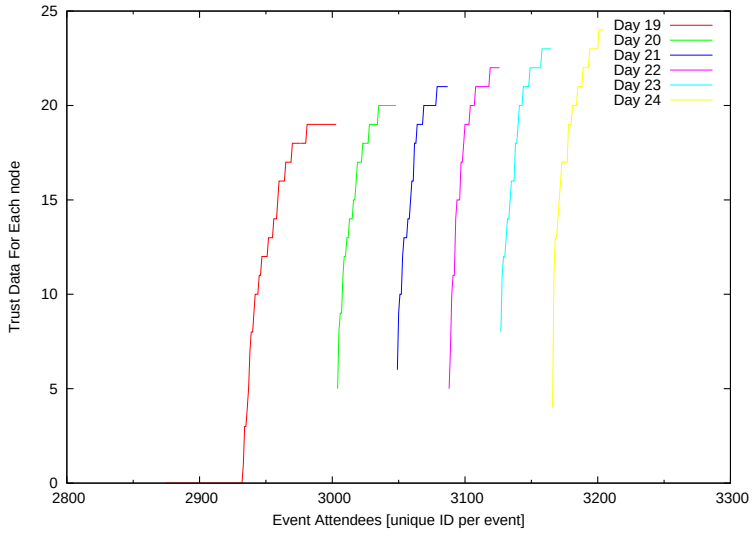


Figure 5.14: Even a careful increase of the trust threshold causes event attendance to drop sharply.

The border line between increasing the threshold too little to get sharp drops, and increasing it and causing sharp drops, is very well defined. An increase of 1.003, which after 27 days lets the threshold increase from 0.5 by a mere 4% to 0.54, shows very similar results to those obtained by no increase at all (i.e. threshold fixed at 0.5), whereas an increase of 1.0038 shows the characteristic drops that we want to avoid. The exact border seems to be around 1.0035 - with this value, I get graphs that are generally very similar to the case of a fixed threshold, but with a slightly yet visibly better damage.

All in all, gradually increasing the trust threshold is an interesting alternative to static thresholds. It requires careful fine tuning of the parameters, but when these are found this scheme equals or surpasses the static threshold scheme. A

particularly interesting feature is that this scheme prevents nodes from denying that they have no trust data in order to be judged by the initial trust value, e.g. if their real trust data are too low for them to receive the message. By increasing the trust threshold and leaving the initial trust static, nodes are forced to show any data they have, otherwise they are not given the message whatever their real trust value. In networks that consist of a fixed set of nodes, so that new nodes don't need to be admitted to it anyway, this is a most useful feature.

## 5.6 Trust with Differentiated Dissemination Behaviour

In this set of runs, some nodes do not act according to their trustees' trust value, and thus the message may end up being disseminated to untrusted nodes. I implement this by choosing a delimiting interest degree to indicate bad disseminators - any node that has a lower interest degree than this limit is a bad disseminator. Hence any node, whether interested in the event or not, may be a bad disseminator. All initial disseminators are set to be good disseminators.

Initially, I choose a interest degree limit of 0.3, which gives me almost 16% of bad disseminators among all nodes (this percentage is computed from the distribution of interest degree, N(0.4,0.1), as defined in chapter 3). The trust threshold is kept at 0.5. Even with such a low percentage of bad disseminators[1] the benefits of my trust scheme are undone. All data collected show exactly the same curves and values as those found in the no-trust runs. Increasing the interest degree limit to 0.4 - causing 50% of all nodes to be bad disseminators - can't affect the data more adversely in consequence, as appropriate simulation results confirm. In neither the 0.3 limit nor the 0.4 limit case does increasing the trust threshold make any difference. Given the significant improvements that an increased threshold yielded for the all-good-disseminators case, this is surprising.

The first setting in which trust showed any effect given differentiated dissemination behaviour is an interest degree limit of 0.26 - 8% of all nodes are bad disseminators - and a threshold of 0.6. Even then, the improvement in comparison to the no-trust case is minimal. After trying several more values, I conclude that any setting with an interest degree limit of more than 0.2 (2.27% of nodes are bad disseminators) reduces the effectiveness of the trust mechanism significantly or completely undoes it. For a limit of 0.18 and a threshold of 0.6, the benefits of the trust value become clearly visible, and reducing the limit further to 0.15 (0.62% are bad disseminators) reduces the average damage to slightly above 0.2, from about 0.25. Even then, the damage isn't ideal - it should be less than 0.2. The fraction of interested reached nodes doesn't drop below 0.8, indicating that a number of malicious nodes has not been detected yet even after a month's time,

[1] low in the light of the mentioned Gnutella study, [14], which speaks of 80% passively malicious nodes in P2P networks

taking into account that more than 20% of nodes have to be excluded from the system to ensure good event damage results.

In general, the percentage of bad disseminators that can't be exceeded without jeopardising the trust scheme is surprisingly low. This makes it particularly important to find a way to counter bad dissemination behaviour, since a fraction of 0.03 bad disseminators, which would already decrease the effectivity of the trust scheme, is very likely to occur, particularly in the light of the mentioned Gnutella study, but also in the light of common sense and an intuition of human behaviour.

## 5.7 Trust with Control of Dissemination Behaviour

Starting with an interest degree limit of 0.3 and a dissemination punishment equivalent to that of moderate misbehaviour, I have simulated the effect of the dissemination tracking scheme described in chapter 3.

The first simulation, with a trust threshold of 0.5, shows no difference at all with respect to the run without dissemination control, equivalently to the no-trust run.

When increasing the trust threshold to 0.7, we can observe the first minor changes with respect to the no-trust runs: After a month's time, the fraction of interested reached nodes decreases by a only few percent. However, the event damage does not change over time. This result confirms what those of the previous section have shown, namely that bad dissemination behaviour has a very strong effect on trust schemes.

To investigate the effect of bad dissemination further I have measured the ratio of bad disseminators to all reached nodes, shown in 5.15. We can see first of all that only about 1% of all reached nodes have bad dissemination behaviour, whereas there should be 16%, since this is the percentage of nodes that on average have an interest degree of less than 0.3.

This is explained to a small extent by the fact that bad disseminators will manage to go to events despite their low interest value and get excluded because of event misbehaviour. A more significant proportion of the decrease is likely to be the effect of bad disseminator that are singled out, i.e. have no contact to other bad disseminators. These nodes collect bad dissemination punishment and are not given the message by their conforming neighbours. A measurement confirmed this explanation: I have counted the number of times that the message is handed from a bad disseminator to another bad disseminator, and contrasted it with the number in which only one of the communicating nodes is a bad disseminator. The ratio of the former to the latter is 0.1 - even assuming that "bad-to-bad" contact are roughly evenly distributed among the bad disseminators, this implies that a large fraction of bad disseminators are singled out after being detected.

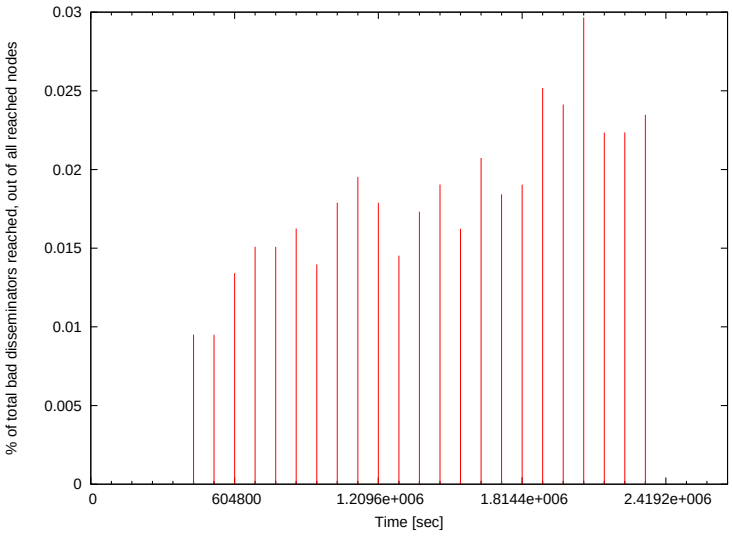


Figure 5.15: The number of reached bad disseminators is lower than their total number in the population, and they get weeded out slowlier than good disseminators.

Those bad disseminators that have others as neighbours are more difficult to single out: First, only bad disseminators on the path of a message that ends up with an attendee can be caught at all, all others may go unnoticed for a long time. Secondly, within clusters of bad disseminators, it is likely that some are not on the path of an attendee, so that even when a bad disseminator has a low trust value, some the other cluster members will be able to hand it the message.

As an aside, note that during the warm up period, no bad disseminators are detected. For the first three days, no node can have more than 3 trust data points, so that the initial trust threshold applies to all nodes, all message exchanges are valid. Also, the proportion of bad disseminators grows since the total number of reached nodes decreases, and the number of bad disseminators is constant.

These results make it seem as if any dissemination control scheme was bound to fail given these settings. To investigate this further, I tried a more severe punishment scheme, by removing any 'positive' entries from the bad disseminators' individual trust data, and adding four 'severe' entries. It is necessary to add at least four such entries, since otherwise the node is considered to have too little trust data and the initial trust value applies. By using this scheme, it is asserted that any detected bad disseminator not in touch with other bad disseminators is excluded from the network immediately, since it trust value is zero (because there are no 'positive' entries). Additionally, the trust threshold is increased to

0.7.

With this setting, towards the end of the month there is a slight decline in bad disseminators in the reached population 5.16 towards the end of the month, however much less significant than is possible with the trust scheme. We can also see that the average misbehaviour of reached nodes as shown in 5.17 declines, which is clear from comparing the third curve in the figure to the others. However that does not correspond to any visible change in event damage, probably because this decline is too small. Possibly, the effects would be more visible if the simulation was carried on for a longer period of time, however the use of such long warm up periods would be questionable in reality, since probably few event organisers would agree to wait for more than 28 events in sequence to ensure good results.

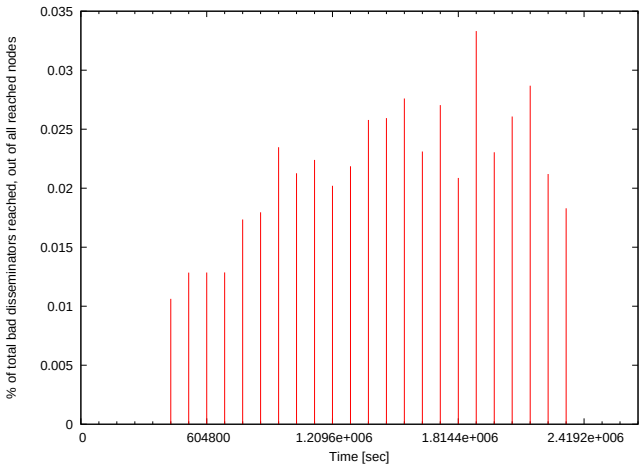


Figure 5.16: The average misbehaviour of reached interested nodes sees a slight drop towards the end of the month

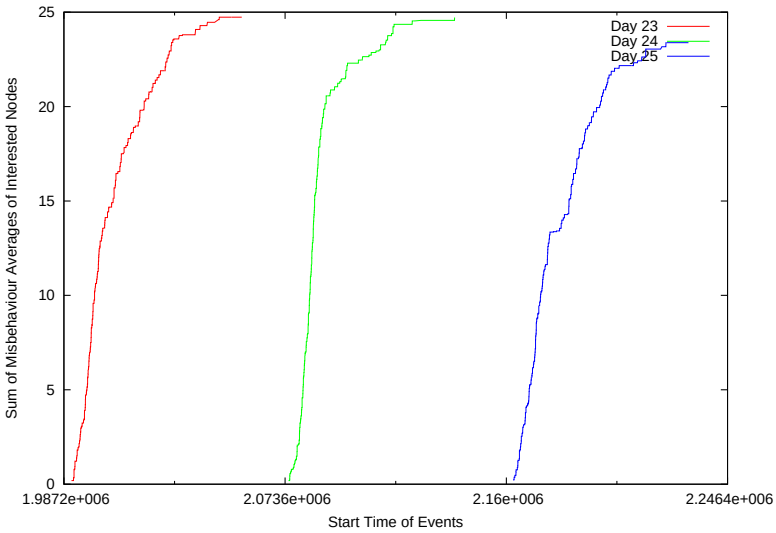


Figure 5.17: The average misbehaviour of reached interested nodes sees a slight drop towards the end of the month. The initial level on day 0 is 25.

This bad result is quite striking given that in the absence of bad disseminators, a trust threshold of 0.7 yields excellent results, and bad dissemination behaviour is punished in the hardest possible way. This is the final hint indicating that trust schemes similar to mine can't handle more than a certain percentage of bad disseminators, since these are then able to form clusters and "collude", which is very difficult to counter.

To back up this idea, I describe several simulations with a lower concentration of bad disseminators in the population to see if from some concentration downwards the dissemination control scheme starts being fruitful.

I start with a low value of the interest degree limit of 0.26 (percentage of bad disseminators is 8%) and trust threshold of 0.7, I also keep the severe punishment of bad disseminators. The data are very close to those that an all-good-dissemination network with the same threshold yielded. In particular, in both cases event damage drops to about 0.13 within a month, as shown in fig. 5.18 for the bad-dissemination case. The fraction of reached interested nodes is lower for the bad dissemination case, it reaches a lower bound of 0.43, whereas without bad disseminators it is about 0.52 (5.19). Clearly this drop is due to bad disseminators that are otherwise well behaved, and would not get excluded from the network if it wasn't for the dissemination control. The fraction of bad disseminators in the reached population drops to zero, see fig. 5.20. Interestingly, the drop is not gradual but very sudden and delayed. This supports the hypothesis

that there must be clusters of bad disseminators, and only when none of then is able to get the message from outside, the members of the cluster are all excluded from the network simultaneously.

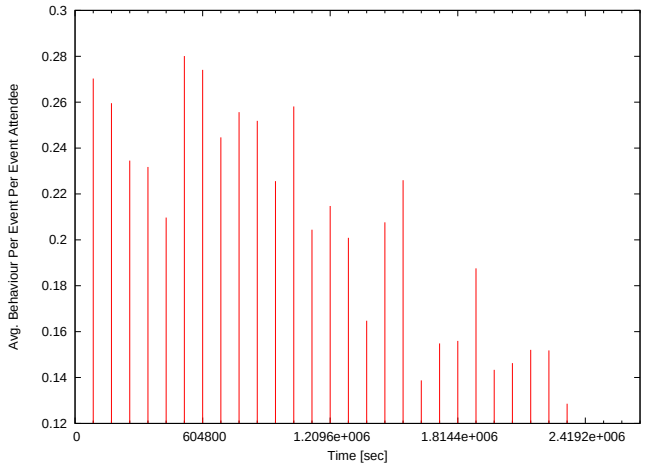


Figure 5.18: Event damage drops drastically when the concentration of bad disseminators in the network is 8%

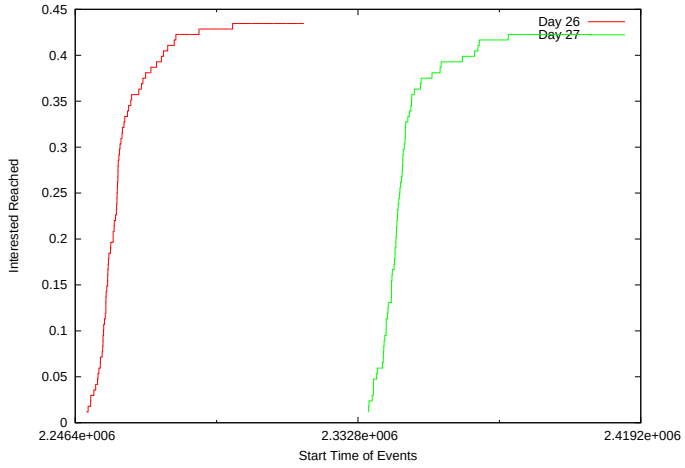


Figure 5.19: For the same setting as above, the fraction of reached interested nodes is slightly lower than in the good-dissemination case

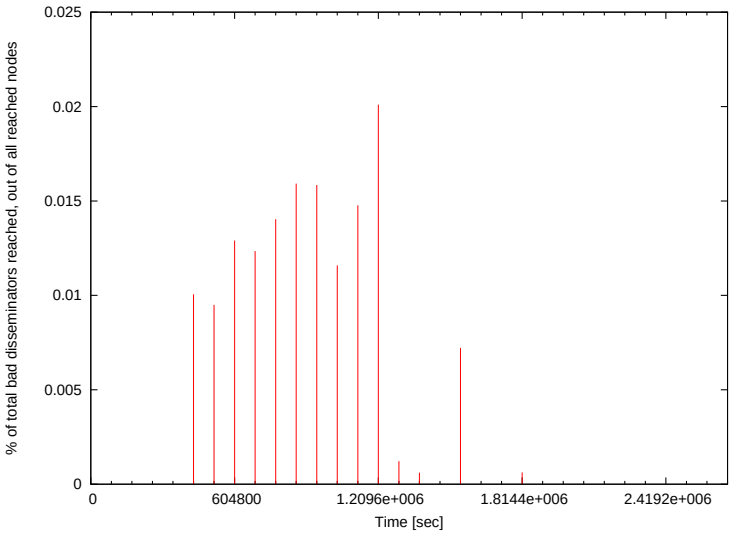


Figure 5.20: For the same setting as above, the fraction of bad disseminators in the reached population drops to zero

It is important to note that nodes are punished immediately when the organiser detects their bad dissemination behaviour, without waiting for them to attend an event at which the trust valued could be updated this implementation. Therefore, it is likely that for the dissemination control to take effect, a longer time period has to go by. I assume that any node attends events at some point, since otherwise it would not join the network, so that any bad disseminator's trust is updated eventually.

For a similar settings, but with a trust threshold of 0.5, the scheme is less effective, with only minor changes in contrast to the no-trust run. The event damage stays slightly above 0.2, and the reached interested misbehaviour only starts decreasing on the very last day. There is a visible effect on the fraction of bad disseminators, but their fraction never reaches zero in contrast to the previous setting, see 5.21. Increasing the threshold back to 0.7, and also increasing the interest degree limit to 0.28 (11.5% bad disseminators) yields good results similarly to the 0.26/0.7 threshold setting. The limit of bad disseminators the network can take is 13.5%, corresponding to an interest degree limit of 0.29. It is then that the event damage stays above 0.2 and that the decrease of the fraction bad disseminators starts becoming smaller.

The last few simulations show that the introduction of bad disseminators requires that the trust threshold is set to more than 0.5, even for small percentages of them, i.e. making the trust scheme less effective. Another insight that is

interesting is that the limit of bad disseminators that may be in the network without jeopardising the trust scheme is not only low, but also well defined.

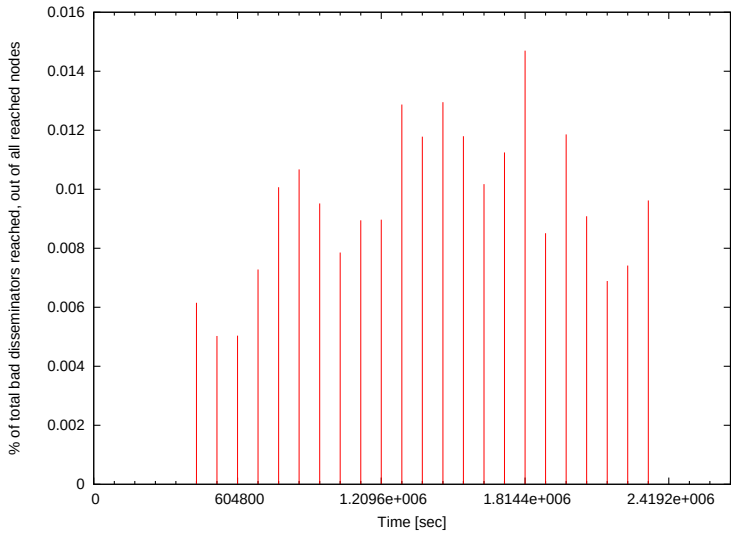


Figure 5.21: The fraction of bad disseminators in the reached population decreases for a threshold of 0.5 given 8% bad disseminators and heavy punishment thereof

## 5.8 Conclusions

After determining that node speed or the number of initial messages does not significantly influence message dissemination, I have run various simulations that show how the trust scheme in conjunction with differentiated user behaviour influences the message dissemination. If all nodes in the network conform to the trust scheme, untrusted nodes are effectively filtered out of the population, while the number of filtered out is not excessively large, and events have full attendance. As an enhancement, the gradual improvement of the trust threshold has been proposed. It yields good results, but the parameters of the network need to be determined with care. With a fraction of more than 13.5% of the nodes being bad disseminators, the positive effects of the trust scheme vanish, irrespective of the trust threshold. This is due to clusters of bad disseminators being difficult to separate from the network.

For an implementation of opportunistic dissemination networks, we can therefore conclude that a trust scheme can be a powerful tool for trust based dissemi-

nation, however it has to be ensured that the fraction of bad disseminators in the network does not exceed a certain limit. This can be ensured in various ways, for instance by incentive schemes similar to those described in [21]. The results from this chapter imply that the lack of effect the dissemination control scheme shows for large percentages is inherent in the network, and so is independent of the trust scheme or dissemination control scheme chosen. It is am important result pointing the researchers of the opportunistic network field at a previously unknown issue that must be resolved prior to any practical implementation of an oppnet that requires a trust scheme.

# Chapter 6

# Conclusion

## 6.1 Summary

In this work, I have implemented and simulated an opportunistic network, in which nodes act as carriers of messages that should only be disseminated to a subset of trusted nodes. The main objective of the work was to investigate and if possible and necessary find a way to counter the expected negative effects of users who do are not willing to cooperate in the trust scheme.

In order to obtain authoritative results, I have decided to create mobility traces that are as realistic as possible, since traces used in much of the related work are known to be unrealistic and therefore invalidate simulation results. The mobility trace creator underlying the traces used here is based on statistical data about human mobility, which overcomes the lack of generality of real mobility models, and is realistic in contrast to the widely used synthetic traces. The traces that it generates are additionally realistic in their spatial model, since it uses Google Maps to determine paths in a real map that take into account many features of the topology such as one way streets.

A trust scheme was designed that is suitable in the given application context, and the relevant aspects of user behaviour were defined.

The subsequent simulations have shown that the positive effects of the trust scheme are annihilated if the fraction of uncooperative disseminators in the node population exceeds 13.5%, even if a dissemination control scheme is in place that detects and punishes a subset of the uncooperative nodes.

The specific context investigated is an event advertisement dissemination network, however the results are applicable for any oppnet in which a trust scheme is to be implemented and the cooperativity of the users with respect to the trust scheme is not guaranteed.

## 6.2 Essential Contributions and Relevance

One of the major contributions of this thesis is the conception and creation of an intuitive and realistic mobility trace. It follows closely the temporal mobility model proposed in [17], but surpasses it in the spatial model. [17] proposes to use GIS or other geographical data to model realistic topologies of cities, but does not provide a realistic way of choosing a path from one location to another. In [9], where a model similar to the one presented here and in [17] is given, the path from one location in the realistic map to another is computed by finding the shortest path using Dijkstra's algorithm, without considering any properties of roads in the map. By contrast, I use Google Maps to determine paths in a real map, which guarantees that these paths are possible and likely. Additionally to being a contribution on its own, using this model ensures that the simulations I run on its basis yield authoritative results.

With respect to opportunistic networks, this is to the best of my knowledge the first work that investigates the effects of uncooperative user behaviour on the effectiveness of a computational trust security scheme. On top of that, I devise a dissemination control scheme to counter the negative effects of the lack of cooperation, and infer from the results of the corresponding simulations the prerequisites that an oppnet has to satisfy in order for any cooperation scheme.

The insight that any dissemination scheme is bound to fail given a fraction of uncooperative disseminators as small as 11.5% already is particularly important result, since it shows that decreasing the fraction of uncooperative nodes is a necessary condition for any trust scheme to work, and that it requires the attention of the oppnet research community before any practical implementation of large scale oppnets can be attempted.

## 6.3 Limitations of this Thesis and Further work

The results obtained in this thesis are based on contact traces using the ONE simulator, which only allows to specify a small set of network propagation characteristics. Therefore, the results should be verified using a simulator that allows to input a large set of realistic parameters (e.g. the well known ns2), so as to verify that the contact traces used here are as realistic as possible. Additionally, the distribution of contact- and inter-contact times should be determined and compared to those of real mobility traces. This is the standard way of assessing whether a mobility trace can mathematically be considered to be realistic, so it would be an important step to do. It was initially planned as part of my thesis, and was not done due to a lack of time.

As a follow on from my work, I obviously suggest to conceive strong incentive schemes for cooperation in trust schemes.

It would also be worthwhile to repeat the work I have done in an oppnet that is supported by a wired backbone, since this is the form of oppnet that is most likely to be implemented in practise, and in fact already has on a small scale, e.g. in [2]. These results would be applicable to existing oppnets, and can be expected to yield different results to the pure oppnet assumed here.

In order to obtain results that would be immediately relevant to a practical implementation of a dissemination network as investigated here, it would be important to repeat and possibly widen the set of simulations done in this thesis using a mobility trace as large as the desired network, for instance using 8000 or 12000 nodes instead of 1800, distributed over a larger city area. This might provide interesting insight whether characteristics specific to the city topology affect the dissemination control and trust schemes. For instance, population density or the topological partitioning of the city because of parks or lakes might influence the dissemination in a way that has not been captured in this work, because the city area considered is relatively homogeneous with respect to these parameters.

# Bibliography


[1] H. Liang A. Srinivasan, J. Teitelbaum. *Reputation and Trust-based Systems for Ad Hoc and Sensor Networks.* Wiley and Sons. [cited at p. 3, 17, 21]

[2] A.Doria A.Lindgren. *Experiences from deploying a real-life dtn system.* 4th IEEE Consumer Communication and Networking Conference, pp. 217-221, 2007. [cited at p. 2, 11, 49, 85]

[3] J. Zupan B. Pushkarev. *Urban Space For Pedestrians.* MIT Press, 1975. [cited at p. 47]

[4] Fan Bai and Ahmed Helmy. *Wireless Ad Hoc and Sensor Networks*, chapter A SURVEY OF MOBILITY MODELS for wireless ad hoc networks. Kluwer Academic Publishers, 2004. [cited at p. 12]

[5] J.L. Burbank, P.F. Chimento, B.K. Haberman, and W.T. Kasch. Key challenges of military tactical networking and the elusive promise of manet technology. *Communications Magazine, IEEE*, 44(11):39–45, November 2006. [cited at p. 1]

[6] Vijay Varadharajan Ching Lin and Yan Wang. Trust enhanced security for mobile agents. *Seventh IEEE International Conference on E-Commerce Technology,pp. 231-238*, 2005. [cited at p. 3]

[7] Vania Conan, Jérémie Leguay, and Timur Friedman. Characterizing pairwise inter-contact patterns in delay tolerant networks. In *Conference on Autonomic Computing and Communication Systems (ACM Autonomics 2007)*, October 2007. [cited at p. 26]

[8] Nathan Eagle and Alex (Sandy) Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http://crawdad.cs.dartmouth.edu/mit/reality, July 2005. [cited at p. 51]

[9] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. Working day movement model. In *MobilityModels '08: Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models*, pages 33–40, New York, NY, USA, 2008. ACM. [cited at p. 15, 52, 84]

[10] Michael Thomas Flanagan. Michael thomas flanagans java scientific library. http://www.ee.ucl.ac.uk/m̃flanaga/java/. [cited at p. 54]

[11] Mike Gavaghan. Java geodesy library for gps. http://www.gavaghan.org/blog/free-source-code/geodesy-library-vincentys-formula-java/. [cited at p. 54]

[12] Andreas Heinemann, Jussi Kangasharju, and Max Muehlhaeuser. Opportunistic data dissemination using real-world user mobility traces. In *AINAW '08: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*, pages 1715–1720, Washington, DC, USA, 2008. IEEE Computer Society. [cited at p. 4, 49]

[13] TKK Helsinki. The one. http://www.netlab.tkk.fi/tutkimus/dtn/theone/. [cited at p. 52]

[14] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on gnutella revisited: the bell tolls? *Distributed Systems Online, IEEE*, 6(6), 2005. [cited at p. 4, 73]

[15] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, Inc., 2001. [cited at p. 24]

[16] Jorjeta G. Jetcheva, Yih-Chun Hu, Santashil PalChaudhuri, Amit Kumar Saha, and David B. Johnson. CRAWDAD data set rice/ad_hoc_city (v. 2003-09-11). Downloaded from http://crawdad.cs.dartmouth.edu/rice/ad_hoc_city, sep 2003. [cited at p. 14]

[17] Vinay Sridhara Jonghyun Kim and Stephan Bohacek. Realistic mobility simulation of urban mesh networks. *Ad Hoc Netw.*, pages 411–430, 2009. [cited at p. 12, 15, 16, 46, 47, 84]

[18] A. Josang. *Subjective Logic*. 2008. Draft. [cited at p. 24]

[19] Audun Josang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and KnowledgeBased Systems*, 9(02184885):279--311, 2001. [cited at p. 17, 24]

[20] Audun Josang and Roslan Ismail. The beta reputation system. In *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002. [cited at p. 22, 24]

[21] Jussi Kangasharju and Andreas Heinemann. Incentives for opportunistic networks. *Advanced Information Networking and Applications Workshops, International Conference on*, 0:1684–1689, 2008. [cited at p. 81]

[22] Thomas Karagiannis, Jean Y. Le Boudec, and Milan Vojnovic. Power law and exponential decay of inter contact times between mobile devices. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 183–194. ACM Press, 2007. [cited at p. 26]

[23] V.Bhuse L. Lilien, Z. H. Kamal. *Mobile and Wireless Network Security and Privacy*, chapter The Concept of Opportunistic Networks and Their Research Challenges in Privacy and Security. Springer Science+Business Media, 2007. [cited at p. 11]

[24] M.Conti L. Pelusi, A.Passarella. *Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks*. Topics In Ad Hoc And Sensor Networking, IEEE Communications Magazine, 2006. [cited at p. 2]

[25] Ching Lin and Vijay Varadharajan. Trust based risk management for distributed system security - a new approach. In *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*, pages 6–13, Washington, DC, USA, 2006. IEEE Computer Society. [cited at p. 4, 16, 19, 22, 25, 26]

[26] Anders Lindgren, Christophe Diot, and James Scott. Impact of communication infrastructure on forwarding in pocket switched networks. In *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 261–268, New York, NY, USA, 2006. ACM. [cited at p. 26]

[27] S. Giordano M. Conti. *Mobile Ad Hoc Networking: The Reality.* Topics In Ad Hoc And Sensor Networks, IEEE Communication Magazine, pp.88-95, 2007. [cited at p. 10]

[28] Stephen Paul Marsh. *Formalising Trust as a Computational Concept.* PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994. [cited at p. 17, 19, 20, 21, 25, 34]

[29] Cesar Hidalgo Marta Gonzales. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008. [cited at p. 12]

[30] Daniele Quercia, Stephen Hailes, and Licia Capra. B-trust: Bayesian trust framework for pervasive computing. Springer Verlag, 2006. [cited at p. 23]

[31] N. Jennings S. Ramchurn. *Devising a Trust Model for multi-agent interaction using confidence and reputation*, volume Applied Artificial Intelligence of *18:833–852.* Taylor and Francis Inc., 2004. [cited at p. 17, 18, 19]

[32] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD data set cambridge/haggle (v. 2006-09-15). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, sep 2006. [cited at p. 14]

[33] Bluetooth Special Interest Group (Bluetooth SIG). Bluetooth core specification v3.0 hs. http://www.bluetooth.com/Bluetooth/Technology/Works/Core_Specification_v30.htm. [cited at p. 52]

[34] Vanessa Davies Tracy Camp, Jeff Boleng. *A Survey of Mobility Models for Ad Hoc Network Research.* Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002. [cited at p. 12]

[35] Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, XXII, April 1975. [cited at p. 54]

[36] Yao Wang and Julita Vassileva. Bayesian network-based trust model. In *WI '03: Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, page 372, Washington, DC, USA, 2003. IEEE Computer Society. [cited at p. 3, 21]

[37] R. Yahalom, B. Klein, and Delta Th. Beth. Trust relationships in secure systems - a distributed authentication perspective. In *In Proceedings, IEEE Symposium on Research in Security and Privacy*, pages 150–164, 1993. [cited at p. 3, 17, 18]

[38] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, 2:1312–1321 vol.2, March-3 April 2003. [cited at p. 12]

[39] J.Vassileva Y.Wang. Baysian network-based trust model. *Proceedings of the IEEE,WIC Int. Conference on Web Intelligence (WI'03)*, 2003. [cited at p. 3, 17, 18, 23, 24]