TAMPERE UNIVERSITY OF TECHNOLOGY

JANNE KULMALA

DEVELOPING LOCAL SOCIAL APPLICATIONS ON MOBILE DEVICES

Master of Science Thesis

Examiners: Adjunct Prof. Marko Hännikäinen
Dr. Tech. Jukka Suhonen
Examiners and topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on 2012-05-09

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
JANNE KULMALA: Developing local social applications on mobile devices
Master of Science Thesis, 56 pages
May 2013
Major: Embbedded Systems
Examiners: Adjunct prof. Marko Hännikäinen, Dr.Tech. Jukka Suhonen
Keywords: Social networks, ad hoc networks, network simulation

Recently online social services have began to utilize the location of a mobile device to find more relevant information for the user. Mobile devices have had capacity for inexpensive wireless communication between neighbor devices, but the technology has not been popular among users due to technical problems and missing applications. This thesis studies possibilities of using direct communication for local social networking.

First, the requirements and use cases are derived for local social appliocations, and an application design is presented which fulfills the requirements. Secondly, a simulation model is described for testing applications that utilize direct communication. The simulation models the movement of a large population in an urban area.

The design was evaluated in a user trial with 250 participants, and the trial participants named local conversations as the most important feature. The simulation model was compared to the user trial and it was found to match the behaviour of the people.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
JANNE KULMALA: Paikallisten sosiaalisten sovellusten kehittäminen mobiililaitteilla
Diplomityö, 56 sivua
Toukokuu 2013
Pääaine: Sulautetut järjestelmät
Tarkastajat: Dosentti Marko Hännikäinen, TkT. Jukka Suhonen
Avainsanat: Sosiaaliset verkot, ad hoc -verkot, verkkosimulaatio

Viime aikoina internetissä toimivat sosiaaliset palvelut ovat alkaneet hyödyntää mobiiliaitteiden paikkannusominaisuutta löytääkseen käyttäjille kiinnostavampaa sisältöä. Mobiililaitteissa on ollut mahdollisuus langattomaan tiedon siirtämiseen lähellä olevien laitteiden välillä, mutta se ei ole ollut suosittua johtuen teknisistä ongelmista ja puuttuvista sovelluksista. Tämä diplomityö tarkastelee suoran tiedonsiirron mahdollisuuksien hyödyntämistä paikalliseen sosiaaliseen verkostoitumiseen.
Ensiksi paikallisille sosiaalisille sovelluksille määritellään vaatimukset ja käyttötapaukset, joiden pohjalta esitetään sovellusprototyyppi. Toiseksi, suoraa tiedonsiirtoa hyödyntävien sovelluksien testaamista varten esitetään simulaatiomalli, joka mallintaa suuren ihmismäärään liikettä kaupunkialueella.
Sovellusta kokeiltiin 250 hengellä käyttäjätutkimuksessa ja tutkimuksen osallistujat pitivät paikallisia keskusteluja tärkeimpänä ominaisuustena. Lisäksi simulaationmallin todettiin vastaavan käyttäjätutkimuksen henkilöiden käyttäytymistä.

FOREWORD

This thesis is a part of research project called "TWIN", and later "Social Index Engine", which took place 2009 – 2013. The project studied the possibilities of local social networking in cooperation between Deparment of Computer Systems at Tampere University of Technology (TUT) and Nokia Research Center (NRC).

I would like to thank everyone who made this thesis possible: My colleagues at TUT, Heikki Orsila, Antti Laine, Marko Hännikäinen, Jukka Suhonen, and people at NRC.

CONTENTS

TERMS AND DEFINITIONS

API	Application programming interface
CPU	Central Processing Unit
DES	Discrete Event Simulator
DHT	Distributed Hash Table
DTN	Delay-tolerant networking
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IRC	Internet Relay Chat
JSON	JavaScript Object Notation
MP4	Video and audio compression method (Motion Picture Experts Group)
NFC	Near-field communication
P2P	Peer-to-peer
POI	Point Of Interest
ROM	Read Only Memory
RPC	Remote Procedure Call
RSA	A cryptographic algorithm (Ron Rivest, Adi Shamir, and Leonard Adleman)
SHA-1	Secure Hash Algorithm
SSH	Secure shell protocol
TTL	Time-to-live
UID	User Identifier
UI	User Interface
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

1. INTRODUCTION

Social services have been one of the drivers for mobile Internet. The largest service, Facebook, has more than a billion active users [13]. Status updates are complementing phone calls and email between friends. Social services are often used to establish new connections and to discover content related to one's personal interests. With new smart phones that have social services integrated into the device, people can access the services anytime and anywhere.

A recent trend [14] is to utilize location information in services, which can turn the service more interesting to particular users. For example, one could find interesting people in same city or suburban area with similar interests. Continuous use of location-based services requires constant Internet connection and positioning (such as GPS). There are challenges involved with this, namely battery lifetime and inaccurary of indoor positioning.

In addition to hosting client applications, mobile phones have capabilities to communicate directly (in a P2P fashion) with neighboring devices using ad hoc wireless networking, such as Bluetooth [35] and WLAN. Inexpensive and relatively fast ad hoc network complements making calls or sending SMSes. Devices can detect and identify other devices in close proximity using wireless broadcast transmissions and by scanning of other devices. New technologies like Wi-Fi Direct [42] are systematizing the interfaces and increasing the performance of media transfers between user devices. Modern phones can also monitor their environment using various sensors [22] to gather context and recognise the user.

Ad hoc communication has been available in the standard IEEE 802.11 WLAN [36] devices since the beginning, but it has not gained popularity among users. In general, three main reasons for this can be identified. WLAN has been a power hungry technology which reduces the operation time between recharging. Secondly, use cases for ad hoc have been missing and no driver application has emerged. The third issue generally raised is the privacy of the users.

Devices often lack the knowledge of their exact location, which poses a problem. Mobile devices today can determine their position using GPS or WLAN access points with the help of an online server. The accuracy of these methods is often limited, especially when used indoors. Better accuracy is needed to determine if two users very close to each other, e.g. in the same room. Another problem is that maintaining a constant connection to the online service consumes energy and reduces the battery life of the device.

Direct communication between mobile devices can solve both of the problems. The devices can communicate in background using an ad hoc wireless radios while conserving energy [1]. When two persons carrying the devices encounter, the devices use the opportunity to discover new information. Each device is carrying personal information that might be interesting to others. The reach of information can be further extended by using DTN, where other devices carry information between disconnected devices [19; 24; 8].

An evaluation method is needed for developing applications that use direct communication between mobile devices. Simulations are often used for investigating the behavior of wireless networks. The reliability of the simulations depend on the accuracy of the artificially created testing scenarios and movement patterns generated for the users. The evaluation of wireless networks protocols are often focused on a single session between devices, while social applications need a model of the whole user population, where each user has its own interests and daily schedule.

This thesis studies possibilities of direct communication between mobile devices. Direct communication is not seen as replacement for centralized infrastructure, but an enabling extension for context-aware services.

The thesis presents the requirements and set of relevant use cases for local social applications. Based on the use cases, an application design is proposed, called Proximate. Proximate implements user profiles, sharing of content, messaging and chat. The application allows extensions with new features. A prototype was developed for the Nokia N900 mobiles device. It uses an experimental WLAN mesh communication implemented by Nokia Research Center [29].

the Proximate application (called the "TWIN") was used to conduct a large-scale user trial at the campus of Tampere University of Technology. The purpose was to collect feedback on how the prototype worked in real life and finding out new use cases [41]. Altogether 250 students and staff members participated in the trial over a period of two months.

As a second main result of the thesis, a simulation model and an architecture is presented for simulating the movement of a large population in an urban movement. The model derives from real-world map information which is analyzed to produce movement patterns that match the behavior of average people. The average person is modeled as having a regular daily schedule and mainly moving between home and the office or school every day.

The mobility model was implemented in a mobility generator that uses map data from OpenStreetMap [7]. The model was compared to recorded mobility behaviour from the user trial.

This thesis is organised as follows. Chapter 2 presents the requirements for local social applications, and in chapter 3 the related work is analyzed based on the requirements. Chapter 4 describes the design of Proximate, a local social application. A simulator design for testing local applications in given in chapter 5. Chapter 6 presents the user trial and evaluation of the simulator. Finally, chapter 7 gives a conclusion of the Thesis.

2. REQUIREMENTS FOR LOCAL SOCIAL NETWORKING

Local communication is little used in mobile social networking, apart from occasional file or business card transfers using Bluetooth. The requirements have been derived for these kinds of applications to gain popularity, diving the analysis to functional user requirements and technology platform performance. This chapter derives a set of requirements for local social applications in general.

2.1 Functional requirements

Autonomous background operation Because of the opportunistic nature of discovering new, short-lived content, the application must collect and analyze the information without user interaction. The occurrence of discovering new, interesting, unpredicatable information is called *serendipity* [39]. The application can combine data shared by other users to previous knowledge. For example, the application can suggest new interesting people or media available at this time and place. With new mobile devices that support multitasking, the application can stay running in the background.

Delay-tolerant networking In delay-tolerant networking (DTN), other devices carry messages between devices that do not have direct connection. When two devices encounter, they exchange the information they are carrying, store the information and forward it to next encountered devices. This type of opportunistic delivery increases the reach of shared information to distant users or large group of users.

Notifying a user when something happens The method of notifying a user depends on the relevancy of the information for the user, which again derives from context (e.g. time, place, other users in proximity). A user should have the control to select (and teach) the application notifications according to personal preferences. Less important events can be collected to a log that can be browsed at a later time.

Discover who is around The user study brought up that following nearby users (friends and unknown) has been the most interesting use case for local social application. User information is stored in profiles that have a structure dictated by the service.

Controlled privacy The user must be able to control the information revealed to other users. For this, a user can use his/her real name, a pseudonym, or stay anonymous with a random ID. Shared information must not leak to unintended parties. Many existing services require that the user gives the real name to discourage misuse.

Two different kinds of communication are common: private between two or more users in a restricted group and public where anyone can follow and participate.

Enough users Large user base tends to attract more users to the service, as many existing friends are already there. Large networks present *social inertia*, which means that the users are reluctant to move to new services, unless enough other users have already switched.

A new application can be made interesting by offering features other services do not have, extending and complementing the existing social services. Easy availability through application repositories makes the application more likely to be installed.

Flexibility for use cases Instead of forcing certain use cases, users should be allowed to come up with new ways of using applications or services. The application can act as a platform on which new ways of social interaction can be built upon, without needing new software.

2.2 Technical requirements

Energy optimization The application should be always on, because the opportunity of finding something is limited. The application should have minimal impact on the battery life of the device.

Every sent message consumes energy on the local device as well as on the receiving devices because the devices have to wake up to process received messages. Especially in mesh networks, sending a message consumes throughput and energy on all the devices that participate to the forwarding.

Privacy Because the traffic in the wireless channel can be received by everyone in reach, the application should make use of cryptography to control privacy. The application should have a method to verify the identity of other users, whether it is their real identity or the identity used in the social network.

Scalability The application must scale up to thousands of active devices at the same time, possible in very crowded area (e.g. a football stadium). The network protocol should be designed so that the amount of traffic needed to maintain the network does not grow exponentially.

Extendability The protocol used by the application should allow new functionality to be added without the need to update the software on every device. When new versions of the application are released, the changes to the protocol must be backwards compatible. Open source is one alternative for evolutionary development.

Large throughput on demand While using optimized communications for autonomous background operation, the application must be able to utilize full throughput for occasinoal media transfers. For example, the full speed of Wi-Fi Direct would be sufficient for most file transfers.

3. RELATED WORK

This chapter summarises the main related work of this thesis on social mobile networking and mobility models for testing social mobile applications. Overall, the topic is new and only few studies exist so far.

3.1 Related work on social applications

The related work consists of reported mobile applications utilizing P2P communication, with the main purpose being social networking. The related work in the area is very versatile, having few common use cases or platform technologies. Many of the related proposals use Bluetooth for ad hoc communication, or mobile Internet access for communication. Security is usually omitted or requires a central trusted party.

Social Net [37] runs in mobile devices and uses wireless communication to detect the presence of other users and suggest new friends. The users are identified by unique identifiers and each user enters a list of friends to the application. Each device maintains a list of unknown identifiers, containing users who have been encountered often but are not marked as friends. The devices exchange the lists of unknown identifiers on encounter.

Hocman [12] is a mobile P2P application designed for social interaction between motorcyclists, such as identifying the encountered bikers at fast speeds or planning for gatherings. Each user has a personal web site stored inside the carried device and the shared files are automatically to copied to the encountered devices using HTTP. The application can be later used to browse the received web pages or discover users that are in proximity. Hocman uses ad hoc WLAN for communication.

PeerHood [33] is a communication system that uses P2P between personal devices on top of a wireless network, such as Bluetooth. PeerHood uses service discovery to maintain knowledge of the immediate neighborhood and provides a method to establish connections between devices. A prototype consisting of a background process and an interface library for applications was presented for a Linux mobile device.

Serendipity [10] performs a periodic Bluetooth discovery to find other devices and records the Bluetooth identifiers of the encountered devices. The identifiers are then sent to a central server where each devices has an associated user profile. A weight can be set for each field in a profile and the weights are used to calculate a similarity score between two users. The users are then notified if the score is above a certain threshold.

MobiSoft [21] is a middleware that uses Bluetooth or ad hoc WLAN to communicate with other mobile devices without a central server. Each device acts as an social assistant that automatically exchanges information with encountered devices on behalf of the user. All information is presented with a unified description format which the devices carry and forward to other devices. MobiSoft evaluates the profiles of encountered users to suggest new potential communication partners and tries to improve the suggestions by learning from the user. The middleware can be also used to share news, private sales or recommendations.

Wireless Rope [28] is a mobile application that periodically scans to gather information of surrounding Bluetooth devices. Repeatedly encountered devices are remembered and certain devices may be marked as contacts. The encounter information is uploaded to a server when the device encounters a stationary device with an online connection. The application displays the encountered devices graphically, where the color represents the familiarity of the device.

WhozThat [3] is a system that connects together online social services and mobile devices and the devices are assumed to have access to the local wireless network and Internet at the same time. The devices periodically advertise the owner's unique identifier on a social network service. Other devices use the identifiers to establish a social context by accessing the user profiles from the online services. One use case for social context is a context-aware music player that adapts the playlist based on the currently present users.

MyNet [20] is a security platform that allows the users to control the access to the services and content of their devices. The owner of a resource can grant a permission to access it by giving out a *Passlet*. A Passlet can give access to a single user, a group of users or a unknown device over an NFC connection. The devices participate in a global P2P network that allows a device to be reached from anywhere. The model of granting access in MyNet matches the behavior of social relationships.

MobiClique [32] is a social networking middleware that uses Bluetooth or WLAN for communication between mobile devices. MobiClique does not rely on central servers and provides an API for third-party applications. The devices exchange application-specific messages on encounter so that the messages travel over multiple devices without infrastructure. Several applications were implemented with the API: Social networking, messaging and distributed newsgroups.

P2Pnet [23] is a communication system for exchanging information in catastrophic disasters or in a battlefield. P2Pnet uses a mesh network consisting of battery-powered WLAN devices, such as laptops and smart phones. As the battery on mobile devices last for hours, P2Pnet can provide connectivity to rescue workers when electricity is unavailable. P2Pnet implements a simple peer discovery on top of standard Internet protocol.

PeerSoN [4] uses direct exchange between devices to solve the concerns of storing profile information in online social services. The devices are connected over the Internet to form a global DHT, which can be used to exchange and store information between the devices. PeerSoN can be used send messages and transfer large files between users that are identified by email address.

Musubi [40] is a social networking application for Android that tries to solve the privacy issues with existing online services. The application only relies on central servers to exchange encrypted information between devices and there is no public information. Two users can establish a connection or new members can be invited to a group by pairing two devices. Musubi allows sharing media, sending messages and reporting location to other users.

Table 3.1 shows a summary of the related work. *Servers* column specifies whether the related work uses central servers in addition to direct communication. *Crypto* column tells if cryptography was used, for example to secure identities. Every work allowed an attacker to track encountered users by collecting the identifiers of the devices. *DTN* column gives whether delay-tolerant networking was used.

Compared to related work, the design presented in this thesis is completely decentralized during use. All information is exchanged using the local wireless connection. Similar to other proposals, such as [33] and [32], our design also provides interface (plugins) for extending the features. Emphasis is on the security of the identities and efficient wireless communication, which are missing from related works. The Proximate design has been evaluated with a large number of trial users, which gives unique results on user acceptance and technical performance.

3.2 Related work on mobility models

The Random Walk Mobility Model is the most simplistic mobility model [5], but is often used due to the good scalability and being easy to implement. The nodes move inside a square area unpredictably by choosing a random direction and a speed. The speeds and directions are changed at constant time intervals or after a certain distance. When the boundary of the area is reached, the node bounces in the reverse direction. To make the movement less random, *Gauss-Markov* model allows the past directions and speeds to influence the future choices. The problem with the models is that nodes are assumed to be independent and do not have any restrictions affecting their movement. Thus, they do not match the behavior of humans carrying mobile devices.

[25] presents a mobility model that is based on the relationships between the per-

Table 3.1: Summary of the related work.

Name	Year	Servers	Communication	Crypto	DTN	Features on social ad hoc networking
Social Net [37]	2002	No	Any (e.g. Bluetooth)	No	No	Learn repeatedly seen strangers, suggest new friends
Hocman [12]	2002	No	WLAN	No	Yes	Share personal web sites, browse the pages of encountered users
PeerHood [33]	2004	No	BT, WLAN, GPRS	No	No	Framework for peer discovery, connections between devices
Serendipity [10]	2005	Yes	Bluetooth, Internet	No	No	Alert of interesting users based weighted profile fields (evaluated at the server)
MobiSoft [21]	2006	No	WLAN, BT	No	Yes	Find new communication partners, share news, recommendations, learn from the user
Wireless Rope [28]	2006	Optional	Bluetooth	No	No	Display history of encountered BT devices graphically, learn repeatedly seen devices
WhozThat [3]	2008	Yes	Any (e.g. Bluetooth), Internet	No	No	Share identity on online social services, context-aware music player
MyNet [20]	2008	No	NFC, Internet P2P	Yes	No	Give users and groups access to personal devices and media
MobiClique [32]	2009	No	WLAN, BT	No	Yes	Delay-tolerant distribution, social networking, messaging, newsgroups
P2Pnet [23]	2009	No	Mesh network	No	No	Communication in catastrophic disasters, peer discovery
PeerSoN [4]	2009	No	Internet DHT	No	Yes	Presence, delayed messaging, file transfers
Musubi [40]	2011	Yes	Internet, encrypted	Yes	No	Secure identity, share media, messages and location
Proximity (this)	2012	No	Any (e.g. WLAN mesh)	Yes	Yes	Secure identity, profiles, chat, share media, messages

sons carrying the mobile devices. Humans tend to organize themselves into closely connected communities. The strengths of the links between the nodes are given as an input to the model, which then clusters the nodes into separate communities. The established communities are randomly associated to a specific square on a topology grid. Each node moves randomly between the squares based on the calculated social attraction for the node to visit different squares. The model does not take into account people interests or limitations imposed by layout of roads and buildings in urban areas.

Working Day Movement Model [11] is a mobility model for delay-tolerant network simulations, where the behavior is largely affected by the distribution of the contacts between the devices. The model captures the behavior of people going to work regularly: They commute to work in the morning, spend the day at work and move back to home in the evening. The model is divided into different submodels defining the behavior of the people in each of the cases. The topology is given as a map containing roads, places and bus schedules. The nodes move between the locations either by walking or using a car or a bus.

Mobile Node Trace Generator (MoNoTrac) [16] is a framework allowing to implement new mobility models based on topology information from OpenStreetMap. The Random Walk Model was adapted to the framework by restricting the movement only to take place along the roads in the map. Another presented model was to pick a random position on the map and move to the position by the fastest route by using a path finder that takes the speed limits of roads into account. The model does not create personal profile for each user, but improves random models by taking urban layout into account.

The recent trend in the related work is to have more realistic models. The models use map information and the road network to model the movement of people between different locations. This thesis combines the idea of daily schedules from Working Day Movement Model to the use of real map information.

4. PROPOSED DESIGN OF A LOCAL SOCIAL NETWORK APPLICATION

In this chapter the use cases and the technical design are given for a local social application, called Proximate. Parts of the design are implemented in the prototype application, and its screen captures are used here for visualizing the features.

4.1 Communication network

Although the networking benefits from energy optimized commnication, our design does not require any particular networking protocol and can be used with a standard Ad hoc WLAN, Bluetooth or other wireless networks. Each device must be able to discover other devices when in radio range. The design does not depend on central servers, nor requires infrastructure WLAN or cellular data commotions when in use.

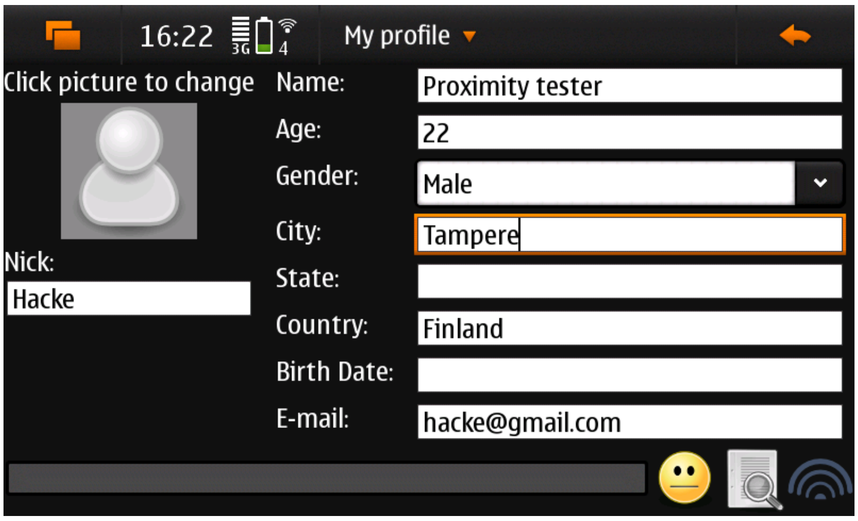
The underlying network can provide a power saving mode. The prototype used an experimental WLAN mesh communication implemented by Nokia Research Center [29].

4.2 Profiles

Each user in the local network has a profile that consists of an avatar and a nick name, and additionally more information, such as address, age, and gender. Users may choose to be completely anonymous. Fig 4.1 shows an example of a profile editor.

To implement peer discovery, each device periodically sends a broadcast message that contains the user identifier (UID) and the profile of the user in encoded form. The application displays the avatars of active users in a view that updates automatically when devices join or leave the network. A user can be interacted with by selecting the avatar of the user.

The profile is a dictionary that consists of key-value pairs and there are no mandatory keys. When the profile needs to transferred or stored, it can be encoded with Bencode [6] or JSON [9] which allow arbitary recursive data structure. This way, the profile can be extended to contain more information when the application gets more features or new modules are loaded to as plugins. Some of the fields have a predetermined purpose, such as the display name and a phone number. In addition



Name:	Proximity tester
Age:	22
Gender:	Male
City:	Tampere
State:	
Country:	Finland
Birth Date:	
E-mail:	hacke@gmail.com
Nick:	Hacke

Figure 4.1: User profile editor in Proximate.

to the information from the user, the profile contains fields added by the application, such as information on shared content.

4.3 Security and identity management

The identities in the network are verified with public-key cryptography, in which each user has an asymmetric key pair that consists of a public key and a private key. The public key can be transmitted to other users over an untrusted network while the private key must be kept secure. The public key is used to encrypt messages, so that only the intended recipient can read the contents. The private can be used to sign messages, and everyone who has the matching public key can verify the identity of the sender. RSA [34] was used as a public-key cryptography implementation, but there are others available.

Every user is identified by a UID, which is a fingerprint of the user's public key, derived from a binary presentation of the key with a one-way hash algorithm (such as SHA-1 [26]). Creating a new key pair that matches a previous UID is practically impossible, which ensures that a UID always matches a single person. The public keys are stored by other devices and automatically distributed to the network alongside the UIDs. In case a user wishes to have the same identity on multiple devices, the private key can be copied over a secure channel, such as using a memory card.

When a device receives a message with an unknown UID, it tries to obtain the matching public key. The request is broadcasted to the network, and can be answered by any device. To avoid thundering herd problem with the replies, the devices wait for a random amount of time before sending the reply. The replies are also sent using broadcast, and if a reply is seen in the network during the timeout, the timers are cancelled to avoid sending any more replies. All devices that have a pending request can benefit from the broadcasted reply.

Using the public-key cryptography is CPU intensive because it involves arithmetic operations with large numbers. For example, the recommended key for RSA is 2048 bits [27]. This is unwanted on mobile devices where processing power is very limited. The problem can be partially compensated by using RSA, where verifying a signature is typically multiple times less CPU intensive than signing a message.

Even with public-key cryptography a problem with ad hoc networks is associating a UID to a person. A malicious user can pretend to be somebody else, because there may be no previous knowledge to verify that users are the persons who they claim to be. Each user needs to personally verify which UIDs in the network can be trusted. The application can provide a button to mark a user profile trusted in the UI, and display the information later when the user is seen again.

A similar verification scheme has been also used in social network services, where the company that runs the service verifies the identities of the users that claim to be celebrities. For example, Twitter and Google+ [15] show an indicator in the profile page when a user has been verified. Also, SSH asks the user whether to trust the public key of the server when the user connects to the server for the first time.

To improve security, the broadcasted discovery messages can be signed with the public key so that a malicious user can not forge the identity of a trusted user. The signature should contain a time stamp so that a malicious user can not repeat the message when the real user is not present.

4.4 Social groups

Social groups are implemented using the user profiles. The groups are identified by arbitrary strings and the user profile has a list of group identifiers that the user belongs to. Anyone can declare being a member of a group simply by adding the identifier of the group to their profile. An identifier can be received from other users or obtained by other means. For example, the web page of a concert could share the identity of a group for people that are going to the concert.

Traditionally in online social services, the first user to create a group becomes the owner of that group, and can decide the members and the topic of the group. Ad hoc network provides no central trust and it is impossible to determine who is the owner of a group.

The groups can have extra information in a profile similar to the user profiles. The information of the groups is maintained in a cooperative way, which means that anyone can modify the profile of the group and distribute the modified version. The profiles are cached in the devices and when a new group identifier is seen in the network, the profile is requested from other devices using the same method as for

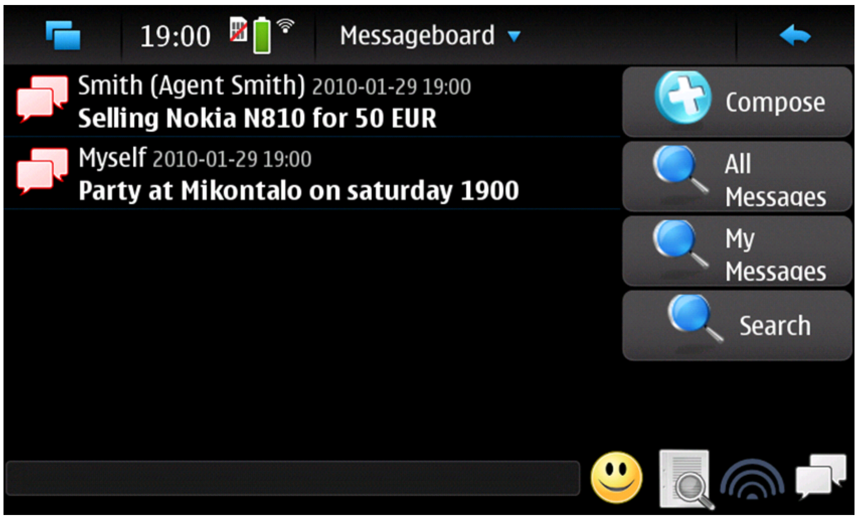

Smith (Agent Smith) 2010-01-29 19:00
Selling Nokia N810 for 50 EUR
Myself 2010-01-29 19:00
Party at Mikontalo on saturday 1900

Figure 4.2: A view that shows messages found from the local community.

public keys. The amount of traffic can be decreased by combining multiple requests and replies to a single message.

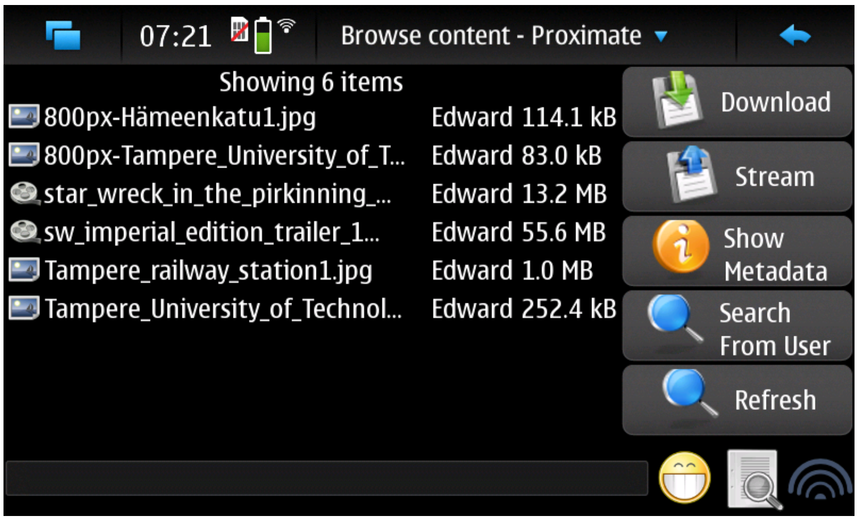
To detect changes, the profiles contain a version number that is incremented when the profile is modified. An existing profile is only overwritten when a profile is received from the network that has a higher version number, and the new version of a profile will eventually be synchronized and replicated to every device. Because the profiles are untrusted, the scheme does prevent a situation where a profile is modified simultaneously in different places. The UI should include a method to disable automatic updates to prevent malicious updates.

4.5 Content Sharing

Figure 4.2 presents the UI view which shows messages from the local community. Each message contains the name of the creator, the composition date and a title. The content of the message is displayed when clicked. On the right hand side are possible actions, such as composing a new message that will be published to others.

In viral distribution, people share content they have created or redistribute what they have found interesting. Content can be e.g. music, videos, pictures, link, advertisement, a memo, a job offer, or any kind of file. Distributing media to a local community also allows podcasting with subscriptions. The local social communications combines the social, virtual presence with geographical proximity of users.

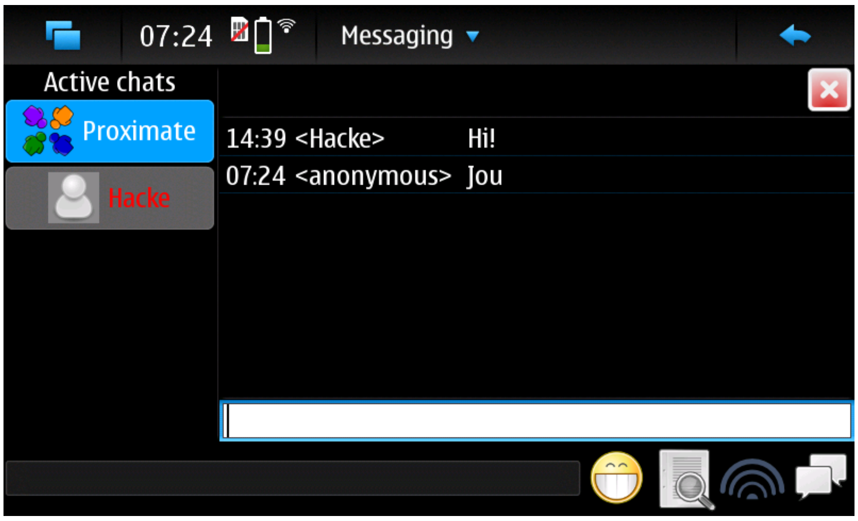
Figure 4.3 presents a view that displays the available files in the network. Each row shows an icon for the file, the file name, the size of the file and the user sharing the file. On the right hand side are actions that can be applied to the selected file. For example, user can download the file to the device or stream the file directly from the other device.



Showing 6 items

800px-Hämeenkatu1.jpg	Edward	114.1 kB
800px-Tampere_University_of_T...	Edward	83.0 kB
star_wreck_in_the_pirkinning_...	Edward	13.2 MB
sw_imperial_edition_trailer_1...	Edward	55.6 MB
Tampere_railway_station1.jpg	Edward	1.0 MB
Tampere_University_of_Technol...	Edward	252.4 kB

Figure 4.3: A view that shows available files in the network.



```
14:39 <Hacke>      Hi!
07:24 <anonymous>  Jou
```

Figure 4.4: A chat view with nearby users.

Figure 4.4 shows an example of a chat with nearby users. The user is having two conversations at the same time: A private conversation with Hacke and group conversation in a group called ”Proximate”.

In the design, share types are unified so that a share can represent a directory, a file, a chat message or a message board entry. The unification gives more freedom for different use cases and content, for example a message board entry can have a picture as an attachment.

Information about a share is stored as a dictionary of key-value pairs called share metas. Typically the meta contains the title of the share and the UID of the creator. In the case a file is attached to a share, the meta contains the file name, the size of the file and information on how to retrieve the contents of the file. The user can enter a list of keywords for the share so that other users can search for relevant

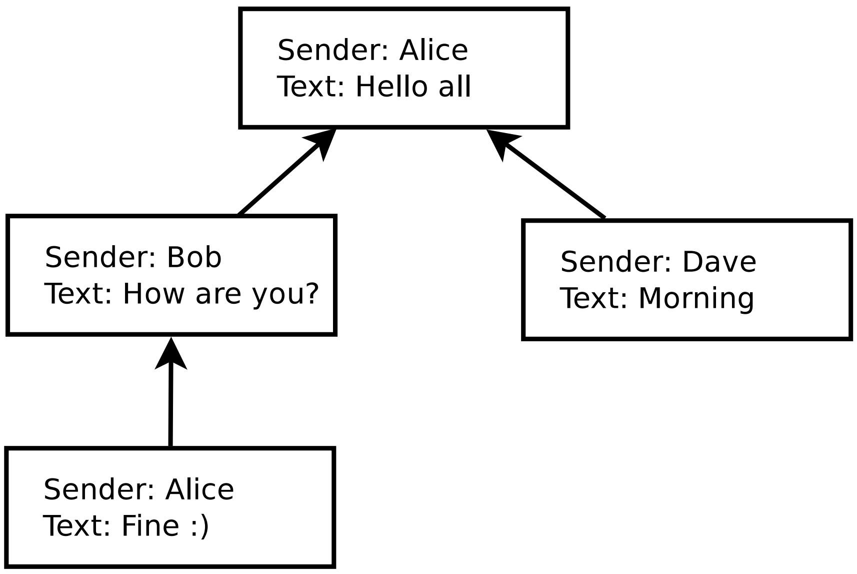


Figure 4.5: An example conversation

shares.

The shares of a single user are identified by numeric IDs and the user profile contains a list share IDs the user has published. When a user profile is transmitted to the network, other devices can retrieve information about new shares by requesting the associated share metas. The devices can cache the metas to avoid duplicate work when a previously met user is encountered again.

The messages can be linked together by adding the share ID of the parent message to the share meta. This can be used to implement conversations, so that a reply is chained to the last seen message in the network. The chains are used to display the messages in the correct order, without relying on having the correct time on each device. When a message is received, the chain can be walked backwards to determine and request missing messages to gather context about the past conversation. Figure 4.5 shows an example conversation with two branches. Alice and Bob have seen the messages from each other, while Dave has answered to Alice's first message.

The share meta can also contain a list of group identifiers for which the share is intended. Other devices can filter shares and only display shares from certain groups. Implementing a more secure way to share to different groups is difficult, because there is no previous knowledge between the devices needed to implement cryptography. Even if cryptography was used, a malicious user could join to an existing group and learn the shared information.

The shares can be replicated and forwarded to a large number of devices using DTN. When two devices encounter, they copy the shares of the other user and begin to advertise the copied shares. Each share has a TTL value in the share meta that is decremented each time a share is replicated to a new device. The TTL value can be used to limit the reach of information to a smaller geographical area. The replication is only suitable for shares without files or shares with a small file, because transferring large amount of data on each encounter consumes a lot of energy.

4.6 Notifications

Events are collected and presented using a unified notification system. Event can either be a query that requires an answer from the user, or a notification that contains information for the user. For example, if the user is invited to a social group, a query is shown. Notifications are collected to a log. Each event contains the UID of the user who caused the event. This way the user profile can be later accessed from the event log, in case an interesting event was noticed after the user has moved away. The events are stored to permanent memory on the device, so that closing the application or running out of power does not result in losing the log.

Each event has a priority. Important events are shown on top of the user interface as pop-ups. Less important are shown on the status bar to avoid distracting the user. The notifications can also include other actions, such as vibrating the device.

4.7 Amount of traffic

The proposed design uses as small as possible number of messages to reduce energy consumption. The peer discovery causes a message to be sent on predefined intervals. The choice of the interval dictates the constant amount of traffic the application causes, even when the environment does not change. A larger interval decreases the traffic but increases the latency to find a new device. The traffic can be minimized by only sending the message when needed, if the underlying communication network supports giving a signal when a device appears. For example, mesh networks usually maintain a routing table and know when devices enter or leave the network.

Encountering a device causes further traffic when either device has published new information after the previous encounter. The devices detect changes in the discovery message and request the profiles of new groups and synchronize the share metas. Each new group and meta cause a request and a reply.

It is common in wireless networks that the overhead of a single packet is large, so it is more beneficial to send a few large messages rather than many small ones. Thus, all the requests and the replies should be combined to the smallest number of message possible. The amount of information that fits in a single message depends on the encoding and the communication network.

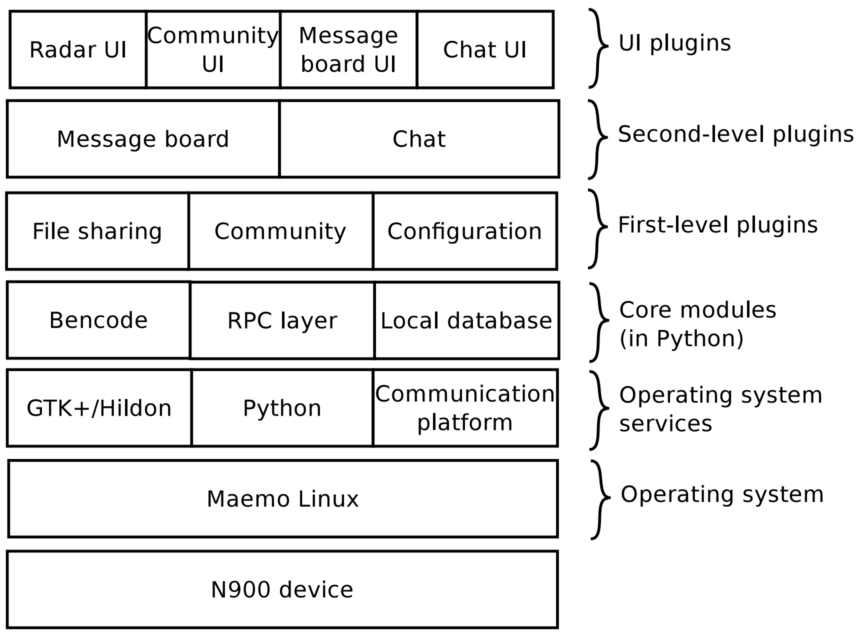


Figure 4.6: The architecture of the prototype.

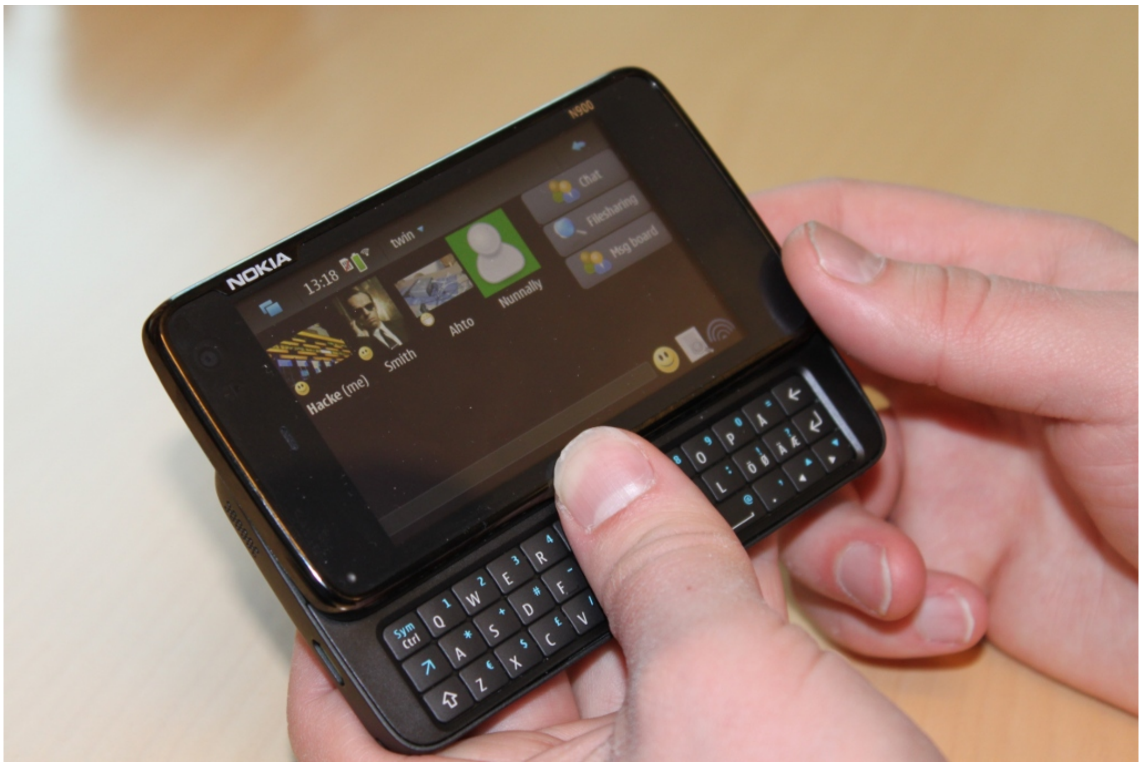


Figure 4.7: The prototype application on a device.

4.8 Prototype implementation

A prototype application was implemented for testing, demonstrations and user trials. The prototype and the underlying communication platform were specifically developed for Nokia N900, which uses Maemo Linux operating system [30]. Proximate was written in Python, using GTK+ [38] for the UI. Figure 4.7 shows the prototype running in a device.

The prototype implementation consists of 13,000 lines of Python code and 1,200 lines of comments in 2,775 commits. More than half of the application is implemented in feature plugins, which contain 7,100 lines. The average size of a plugin is 400 lines, file sharing being the largest plugin with 1,300 lines. The smallest one is a plugin that vibrates the device when a friend appears in the network, and consists of 44 lines. The application size, including graphics, is 1.4 MB.

Figure 4.6 presents the architecture of the prototype. Each module in the diagram uses the services of the layers below the module. The application consists of core modules and a number of plugins, which can offer an interface for other plugins to use. The first-level plugins implement core functionality that the more specialized second-level plugins are build up on. All user interface-related functionality is in separate plugins which interface with first-level and second-level plugins. This allows the user interface of the application to be changed without rewriting the underlying functionality.

Public-key cryptography was not implemented in the prototype. Instead, the UIDs were chosen randomly. The prototype consists of the following modules and plugins:

Bencode A module to encode and decode information using Bencode. All communication in Proximate uses Bencode, which is a compact serialization format for transferring and storing recursive data structures.

RPC layer Offers an RPC interface to hide the details of the underlying communication network. Other modules can perform requests to a single device or a group of devices or register as request handlers. The requests can either be performed as reliable or unreliable. For reliable requests, the remote device sends an acknowledgement message and unsuccessful requests are automatically retried.

Local database Data for the plugins can be stored to and retrieved from the local database. It implements a high-level object interface which allows plugins to store Python data types on the disk. Other modules can register new object types and provide a schema which is used to validate the objects.

File sharing A core plugin which implements sharing information to the local community, including retrieving share metas, transferring contents of shared files and DTN of the shares. The plugin contains both the server and the client side for transferring files, and provides a method to store downloaded files or stream them directly using the media player of the device. Figure 4.3 presents a screen shot of the file sharing UI.

Community A core plugin containing user management and peer discovery. This includes storing and tracking active users and social groups in the network. Other modules can subscribe to events about changes in the network.

Configuration Implements user preferences and an interface for plugins to read and save configuration settings that are stored into the configuration file.

Community UI Provides UIs for browsing users and social groups, creating and deleting groups, and viewing and modifying user profiles. Uses the services of the community plugin.

Message board and Message board UI Contain the additional functionality to implement a message board on top of the file sharing plugin. The message board can be used for local announcements, questions or any relevant notices. The message board UI plugin implements UIs for composing and browsing messages.

Chat and Chat UI Implement distributed private and public conversations. In the prototype, the chat was implemented as a separate plugin, but could be instead written to use the share architecture. The UI plugin implements a UI similar to Internet chats and the ability to switch between different conversations using tabs.

Radar UI Visualizes the currently active users in the local network. Uses the community plugin to access information about the current state. Figure 4.8 presents an example. The view is updated when users enter and leave the network. The rings represent the number of routing steps needed to reach other users in the mesh network.

The users in the innermost ring are one hop away and can be reached directly. The users in the second ring are in two hops distance, meaning that the devices are outside the reach of a radio signal and the traffic is routed over other users. The number of hops can be used to estimate the physical distance to the user. For example, assuming that the reach of a radio signal was 50 meters on average, the users in the second ring would be 50–100 meters away. The distance also indicates

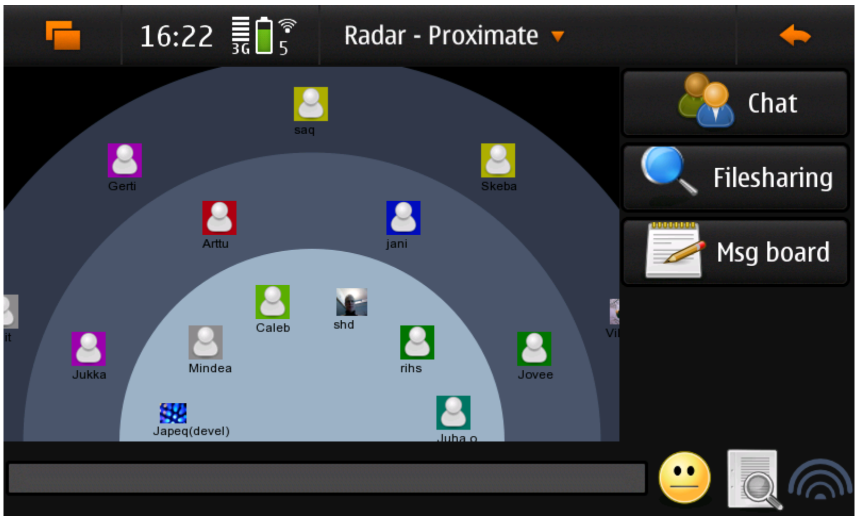


Figure 4.8: The radar view showing present users.

the unreliability of the connection to the other users because routing traffic over multiple devices in a dynamic environment increases transfer errors.

5. SIMULATOR FOR LOCAL SOCIAL NETWORKING

This chapter is a description of a simulation model and a simulator architecture which are designed to be used for large-scale testing of social mobile applications.

5.1 Simulation flow

The simulation flow consists of different programs that read input from a file and write output to a file in a format that the next program can process. The parameters and the immediate files are stored in a disk and encoded for example with bencode or JSON. This way, it is possible to implement the programs in different languages or write new programs to analyze the data. Also, testing individual programs is easier. Any part of the flow can repeated multiple times in a reproducible way and the immediate files can be archived for later examination.

The simulator provides a common interface, called Client API, so that different applications can be tested without modifying the simulator. The API gives state of simulated world from the point of view of a single device and provides means for the application to communicate over the simulated network.

Figure 5.1 presents the flow of the simulation model. The direction of the flow is downwards. The left side presents the components that are specific to the application. Because each application depends on different social information, the generation of social information can not be generalized. The components on the right belong to the simulator. The most important components in the figure:

Map A vector map of the urban area where the devices are moving. For example an XML map from OpenStreetMap.

Mobility generator A program that implements the mobility model. The generator takes the map information, analyzes it and writes movement patterns for all devices into a file.

Mobility information Contains the movement patterns of the devices for the simulator. The file can be be either generated with the mobility generator or with any random mobility model, because it does not contain the map.

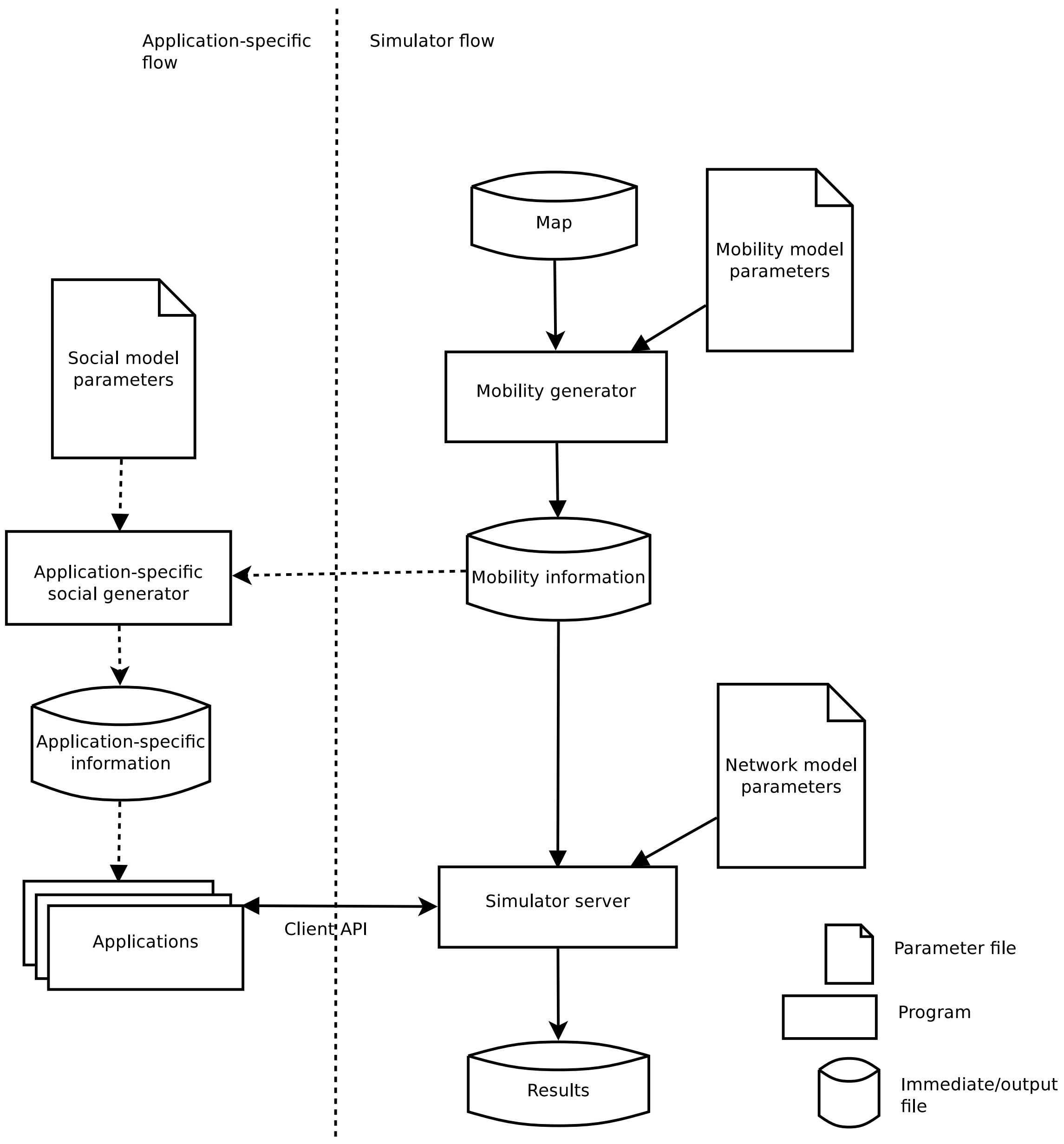


Figure 5.1: The flow of the simulation model.

Simulator server A program that takes the mobility information and executes the simulation. The applications and the server are connected together via Client API.

Application-specific social generator Program that generates social information for the application. The program can optionally use the mobility information if the social model depends on the locations of the devices.

Application-specific information Input for the application being tested, for example user profiles or behaviour patterns.

Applications One or more applications being tested. The applications do not necassarily have to use input. Applications can report various statistics via Client API, such as how succeful the application was or what kind of impact it had on the user.

Results The simulator server generates a results file that contains statistics about the network traffic and the application and allows correlating between them.

5.2 Simulator architecture

During a simulation, the simulator server and the applications are in different processes which are executed simultaneously. The applications connect to the server by using Client API which is implemented over UNIX or TCP sockets. Having separate processes prevents accidentally leaking state between the applications and allows the applications to written in different languages.

UNIX sockets are used for communication between different processes that are on the same machine, while TCP can be used to run the simulation on several machines connected to LAN or Internet. This allows better scalability and the simulation can utilize multiple CPUs and machines without any changes. This way, the simulation can scale up to thousands of simulated devices by using a cluster of interconnected machines. Figure 5.2 presents the architecture.

The server is a discrete event simulator (DES) which models everything that happens in the simulation as a sequence of events on a timeline [18]. The server maintains the state of the simulation, which consists of the simulation time, positions of the devices and state of the network. Whenever the application needs to obtain the current time or schedule an event into the future, it uses the API instead of the operating system. This allows the simulation speed to be faster or slower than the wall clock. All events have to be known advance, but in turn it allows the simulation to quickly step from an event to the next.

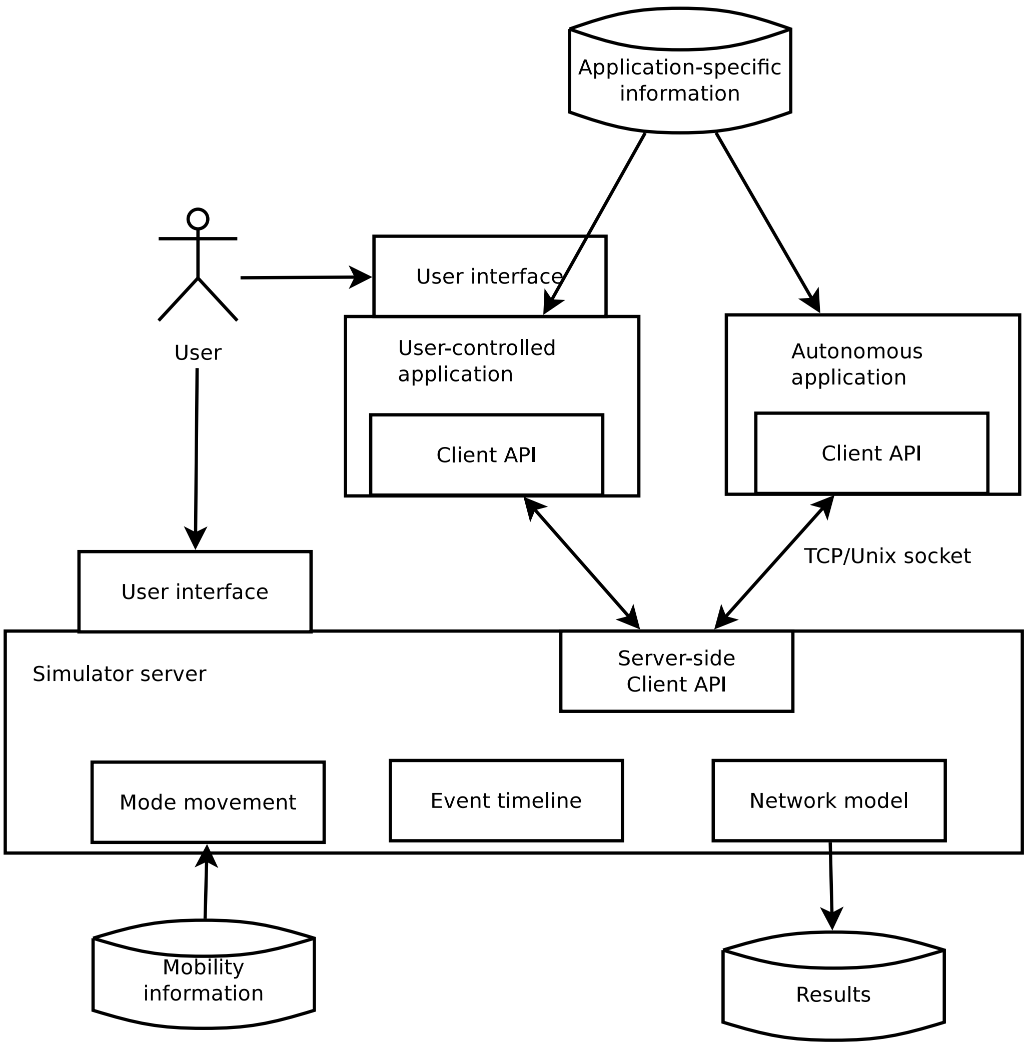
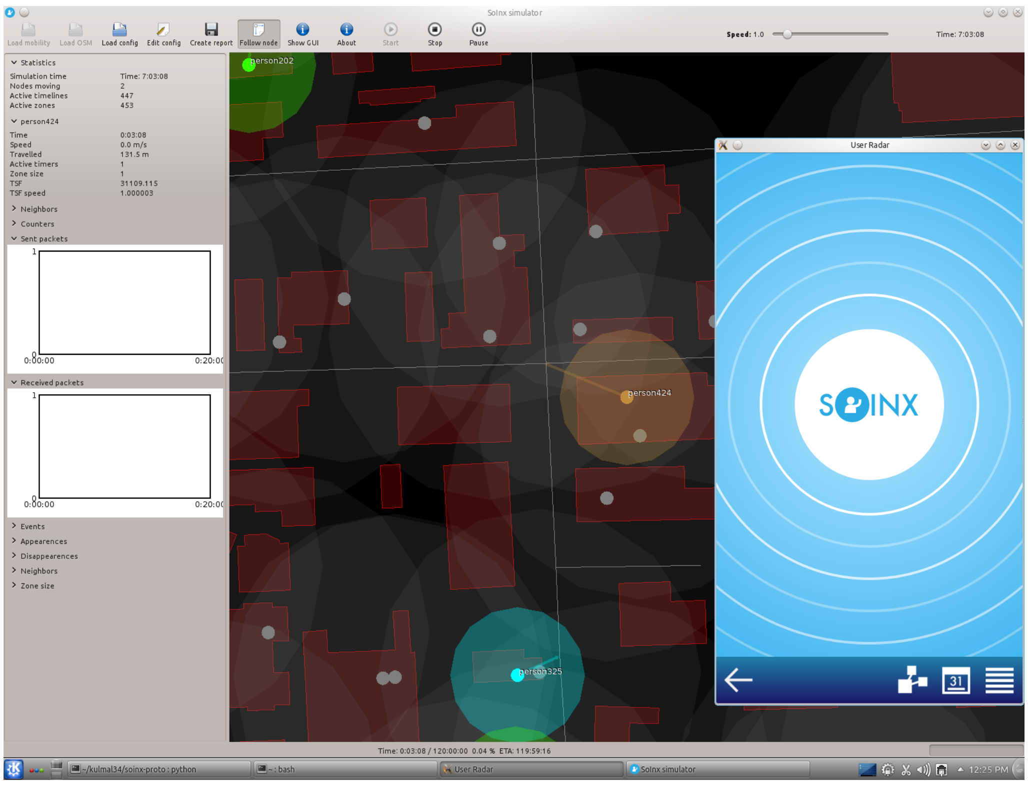


Figure 5.2: The architecture of the simulator.



Simulation time	Time: 7:03:08
Nodes moving	2
Active timelines	447
Active zones	453
person424	
Time	0:03:08
Speed	0.0 m/s
Travelled	131.5 m
Active timers	1
Zone size	1
TSF	31109.115
TSF speed	1.000003

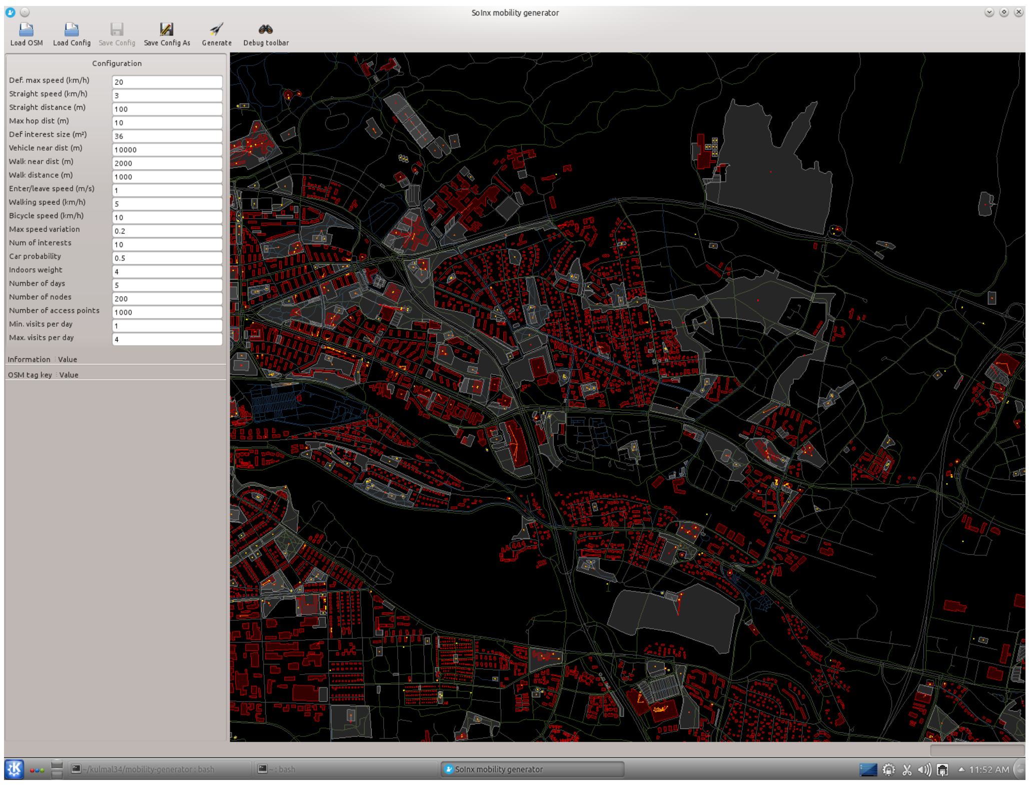
Figure 5.3: The user interface of the simulator.

The server has a user interface that shows the state of the simulation and a view where the devices are located on the map. The applications can optionally have user interface so that the simulation be tested with real users. Figure 5.3 shows a screen shot of a simulation. The panel on the left shows statistics on the selected device, and the view on the center shows positions of the devices on map and the reach of radio. The smaller window on right is the UI of an application connected to the simulator.

Figure 5.4 shows a screen shot of the mobility generator. The panel on the left is used to modify the parameters of the mobility model, and the view on the right shows the currently loaded map. The map view can be used to find problems like missing weights or errors in the path finder.

5.3 Client API

The Client API has methods to send and receive packets from the network, similar to what a real device would have. The packets are treated as arbitary binary strings. The applications should not exchange information directly outside the API and the API is designed to discourage this. Also, the applications can report statistics on how succeful the application was, and all statistics are included in the results file.



Def. max speed (km/h)	20
Straight speed (km/h)	3
Straight distance (m)	100
Max hop dist (m)	10
Def interest size (m²)	36
Vehicle near dist (m)	10000
Walk near dist (m)	2000
Walk distance (m)	1000
Enter/leave speed (m/s)	1
Walking speed (km/h)	5
Bicycle speed (km/h)	10
Max speed variation	0.2
Num of interests	10
Car probability	0.5
Indoors weight	4
Number of days	5
Number of nodes	200
Number of access points	1000
Min. visits per day	1
Max. visits per day	4

Figure 5.4: The mobility generator.

5.4 Mobility information

The mobility information of a single device is presented as path. A path is a series of tuples in the form *(Time, Position)*, *Time* is a time and *Position* is a 2D coordinate. During a simulation, the position and the speed of a device can be calculated with linear interpolation between two subsequent track points. Before the first point in a track the device is stationary at the position of the first point and after the last point in a track the device stays at stationary at the last position.

Figure 5.5 shows an example of the tracks that two devices will travel. The devices will become in proximity when the distance between the devices is less than the radio range and distant when they move outside the range. Because all events have to be known in advance, the exact time when the tracks will collide are calculated in the beginning of a simulation.

5.5 Network model

5.5.1 Multi-hop ad hoc network

The network model is an high-level abstraction, which means that it estimates the radio channel and network protocol using a model that behaves statistically similarly. Using an abstract model makes the simulation much faster than using very accurate

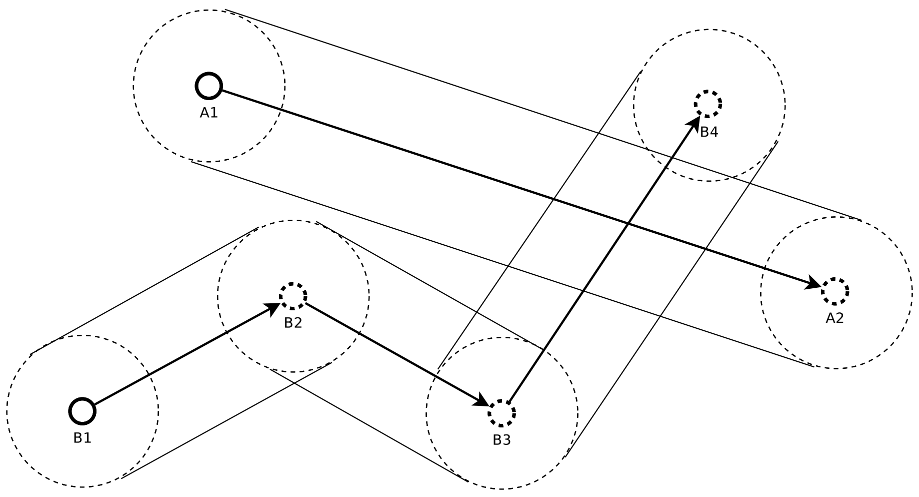


Figure 5.5: An example mobility scenario.

model and allows scaling up to thousands of devices.

The model is based on ad hoc mesh network which incorporates power-saving. The network automatically formed between proximate mobile devices and other devices participate in forwarding traffic by re-transmitting packets. This allows packets to reach devices that are outside of the radio range of a single device. Packets are sent to the network by flooding in which all devices in the network participating in forwarding the packets to maximize coverage and reliability of the transmission. Figure 5.6 shows an example of a multi-hop ad hoc network.

5.5.2 Packet loss model

Unreliability of a radio channel causes packets to be corrupted or lost during transmission. Unreliability is caused by the background noise, transmissions from other devices and the decay of the signal over long distances.

The packet loss is modeled by probability of lost packet over a direct radio channel. Because modeling a radio channel is difficult problem, a linear model is used instead which is easier to configure experimentally.

The packet loss is calculated as follows:

$$P_{\mathrm{dist}}(d) = \begin{cases} (P_{\mathrm{rel}} - 1)\frac{d}{d_{\mathrm{rel}}} + 1 & \text{if } d < d_{\mathrm{rel}}, \\ (0 - P_{\mathrm{rel}})\frac{d}{d_{\mathrm{range}} - d_{\mathrm{rel}}} + P_{\mathrm{rel}} & \text{if } d > d_{\mathrm{rel}} \wedge d < d_{\mathrm{range}}, \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

where d is the distance between the devices, d_{range} is the maximun reach of a transmission, and d_{rel} is distance where the probability is P_{rel}. Figure 5.7 shows the packet loss as a function of distance. The constants can be chosen so that with transmission distance shorten than d_{rel}, the packet loss is minimal, but after d_{rel} it

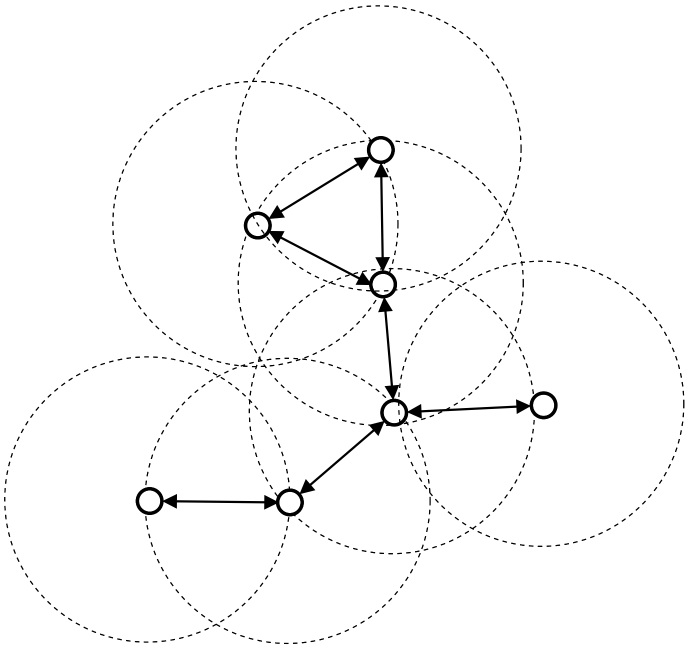


Figure 5.6: A multi-hop ad hoc network between mobile devices.

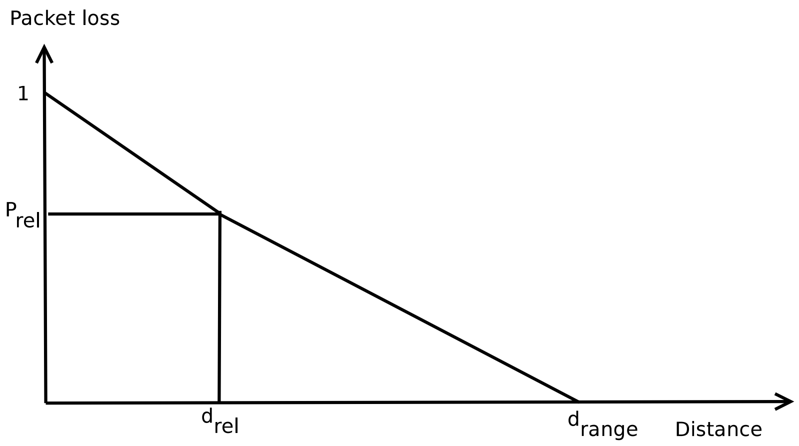


Figure 5.7: Packet loss as a function of distance.

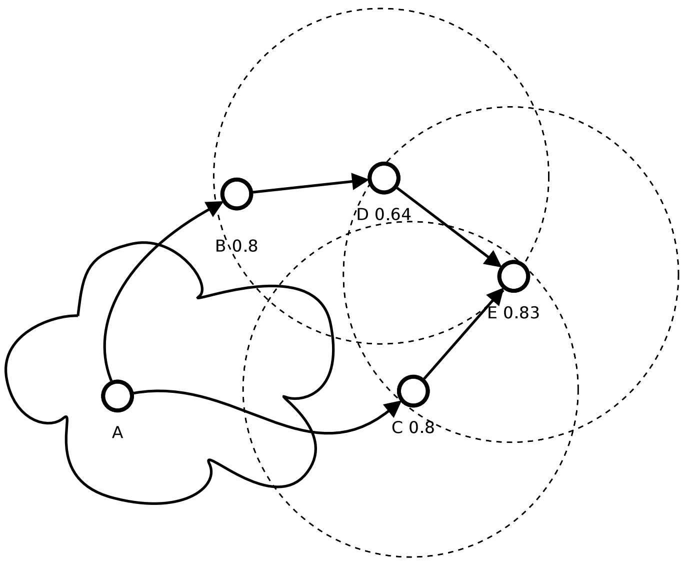


Figure 5.8: An example of multi-path packet loss.

starts to grow fast. Finally, the transmissions are not possible at all over distance d_{range} due to the signal disappearing into the noise.

5.5.3 Multi-path packet loss

Because packets are transmitted over multiple radio channels, the packet loss depends on the number of forwarding steps and different paths between devices.

Figure 5.8 shows an example of a multi-path packet loss. device A is transmitting a broadcast packet to every other device. There are two path over which the packet can reach device E, which increases the reliability of the transmission.

The packet loss over multiple paths can be calculated using inverse probability, as the events that the packet reaches the neighbor devices can be treated as being independent. The probability of a packet reaching device N is as follows:

$$S(N) = 1 - \prod_{x \in M} (1 - S(x)(1 - P(N,x))) \tag{5.2}$$

Where $P(x, y)$ is the packet loss between devices x and y, $S(x)$ is the probability of a packet to reach device x and M is the set of devices that are in the route. Because this assumes that the probability for the route is known, the probability must be calculated for each device starting from the sender and going outward.

5.5.4 Latency

Devices are asleep as much of time as possible to save power and the devices wake up periodically to send and receive traffic. All the devices that wish to communicate with each other must be awake at the same time in order to receive and transmit

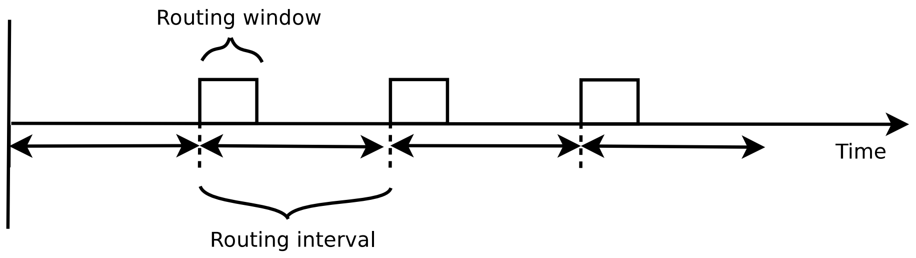


Figure 5.9: An illustration of the power save cycle.

packets. The devices follow a periodic cycle where the wake up periods of the devices are syncronized so that a device that wishes to transmit a packet can predict the time when the other devices are listening to the radio channel.

Figure 5.9 is an illustration of the power saving cycle. The duration during which the devices are awake is called the routing window.

A packet is forwarded one step further on each routing window. The latency of a transmitted packet to reach a device is determined from the shortest path from the origin to the device, or the smallest amount of forwarding steps. Dijkstra's algorithm can be used to determined the shortest path to every device in a zone.

The simulation time of a delivery of a transmitted packet to a device is calculated as follows:

$$t(N) = t - (t + t_{offset}) \text{ MOD } t_{interval} + t_{interval} hops(N) \tag{5.3}$$

Where N is the receiving device, t is the current simulation time, t_{offset} is the offset of the routing cycle, $t_{rinterval}$ is the routing interval and $hops(N)$ is the smallest number of forwarding steps from the origin to the device N.

5.5.5 Communication zones

Communication zone is the set of devices that follow the same power saving cycle. The zones are as large as possible in order to the traffic to reach the maximum number of devices.

Figure 5.10 shows an example of a zone change. In the beginning the two devices are distant from each other but move in to radio reach after a while. The devices will not see each other instance, because they are in different cycle. There may be two or more overlapping zones in the same place due to movement of other devices.

A device sometimes stay awake during the whole routing interval cycle to be able to find devices in other zones. If a device detects a larger zone than the zone it currently belongs to, it tries changes to the new zone. The neighbors of the device notice the zone change after a delay, and change to the larger zone as well until all nodes have migrated to the largest zone. Figure 5.11 shows the merge of two zones.

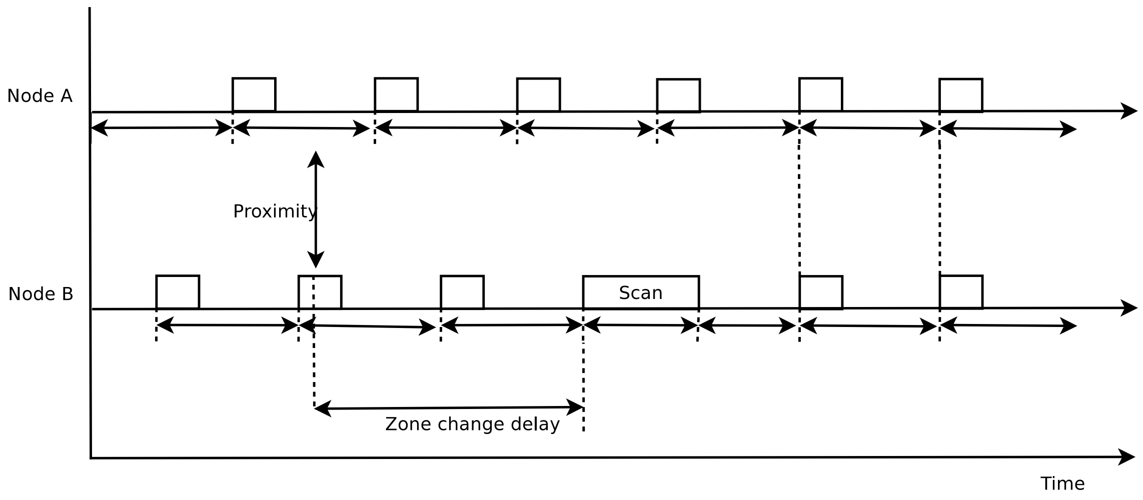


Figure 5.10: An example of a zone change.

5.6 Mobility model

Figure 5.12 presents the steps during the creation of mobility information. Map information is first preprocessed so that it can be used for the mobility model. The model then creates a profile for each simulated person based on the map information and weights from a configuration file. As the simulation can consists of multiple days, a set of activities is chosen for each person on beginning of each day. Then, a travel plan is created for each person for the day. Finally, the resulting mobility information is written to a file, which can used in a network simulator.

5.6.1 Map information

The mobility model requires a vector format map where all positions are represented by map nodes with a geographic longitude and latitude coordinate. A node can optionally have extra information attached, if it represents a point that has relevancy in the map. Such positions are called points of interest and are used to point the locations of businesses, landmarks, etc. The nodes are also used as the vertices of polygons representing relevant areas larger than a single point. Each polygon has an associated type and the polygons can be used to describe buildings, sites, parking areas, parts of a city and other features.

Roads and other travel paths are represented as polylines, which consists of the map nodes. The type and the speed limit of each road is given as extra information. The roads must be connected together by common nodes to form a road network, which can be used to find paths between different location.

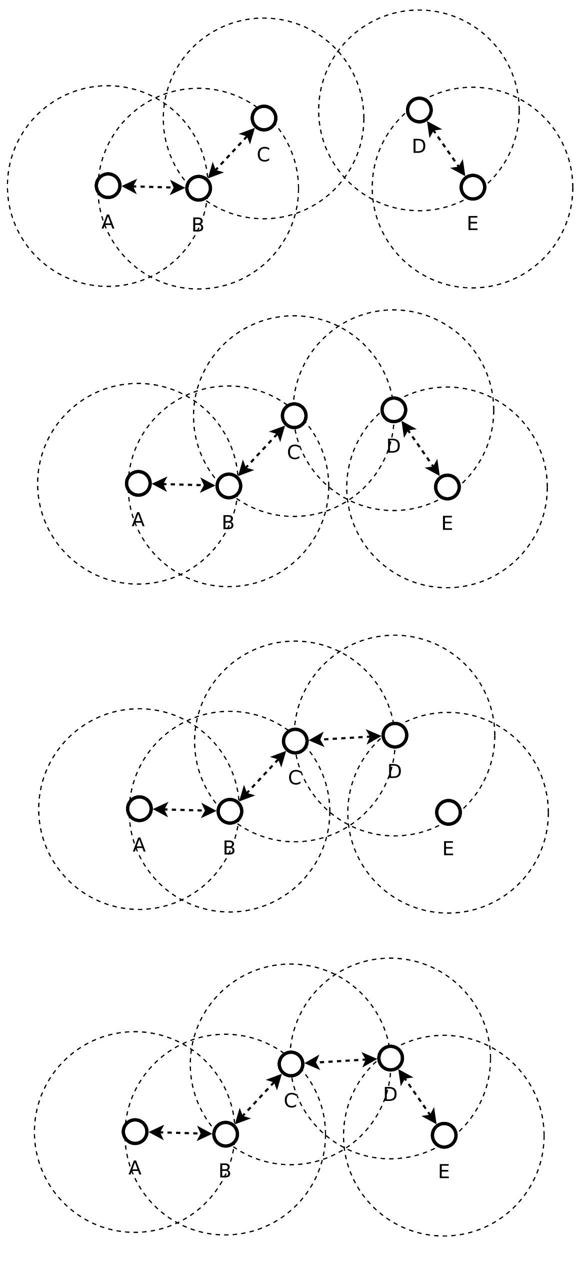


Figure 5.11: Two communication zones merging.

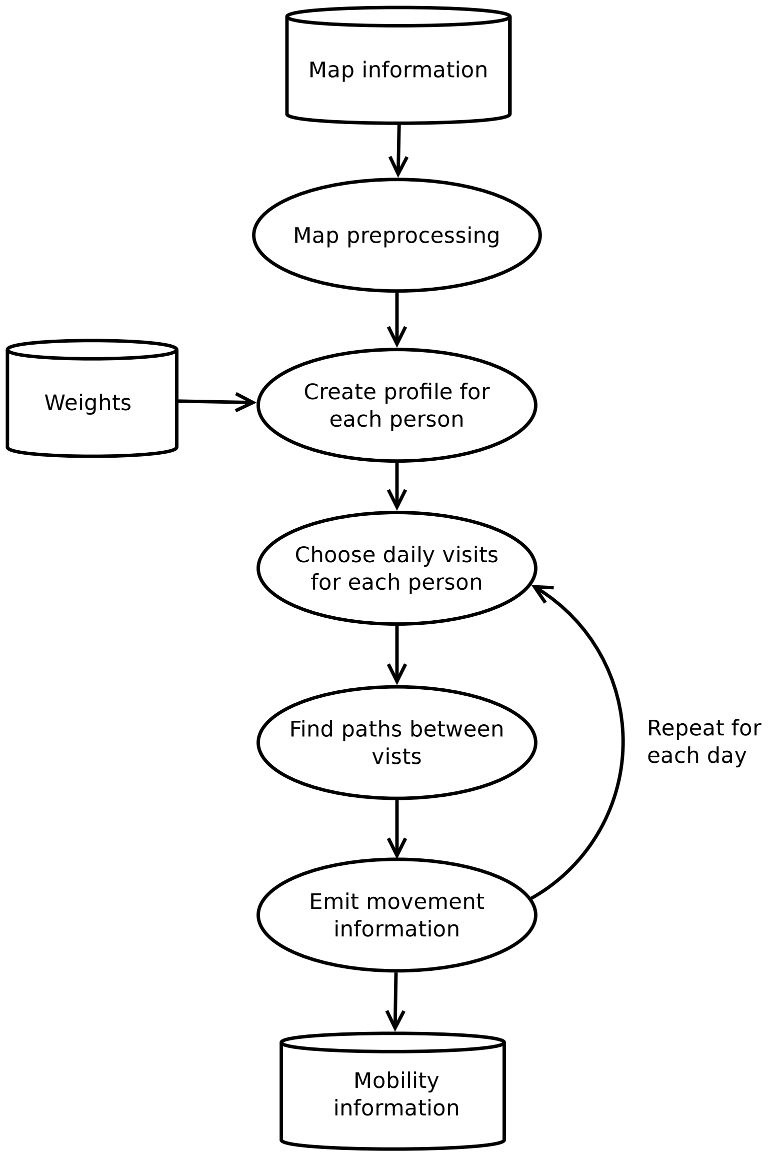


Figure 5.12: Steps during the creation of mobility information.

5.6.2 OpenStreetMap

The prototype uses XML map information from OpenStreetMap. As of 2012, the project has half a million users, who have entered over a billion coordinates [31].

Because OpenStreetMap is updated by volunteers, the map tends to be very detailed in dense urban areas while having only the main roads in less populated areas. As we are interested in simulating movement of devices in urban areas, the map information is sufficient for the prototype.

5.6.3 Map preprocessing

The map information is preprocessed to be more suitable for path finding and creating profiles for the simulated persons. All geographic coordinates in the map information are translated to a two-dimensional plane where the axes use the same scale, e.g. 1 meter. This allows the use of simple vector arithmetic for faster computation. The polygons are estimated with bounding boxes. The optimal bounding box can be obtained by fitting boxes with different rotations around the polygon and choosing the box with the smallest area. For each polygon with interest information, a matching POI is created at the geographic center of the bounding box.

The bounding boxes are used to determine the polygon to which each POI belongs to. This is done because POIs often point places that are inside a larger area, and the models needs to know which POIs belong to which polygon. For example, shops placed inside a building are treated as being inside the building. Bounding boxes allow some degree of human error in case there are POIs that should be inside a polygon but are placed outside e.g. a concave polygon. Each POI is treated to belong to the smallest polygon that is at that position, because there are often overlapping polygons. For example, in case a building is part of a larger property, the POI is connected to the building. Also, the polygons representing city borders and bodies of water are ignored in this step.

5.6.4 Simulated persons

The mobility model creates personal information for each simulated device using the map information. For each person, a home is randomly selected from the all buildings in the map, while ignoring buildings that have interest information, such as shopping malls and office complexes. A weight is assigned for each building, and the probability of choosing a building is the weight divided by the sum of weights of all buildings. The weight of a single building is

$$W_{\text{building}} = \frac{A_{\text{box}}}{n_{\text{building}}}, \tag{5.4}$$

where A_{box} is the area of the bounding box that represents the building, and n_{building} is the number of interests that are inside the building. Thus, larger buildings are preferred because they tend to have more people living in them, while a number of POIs decrease the amount of living area. The exact position of the home inside the building is randomly chosen from uniform distribution inside the bounding box of the polygon.

The simulated persons represent average humans who have a single important place the go to on daily basis, such as an office or a school, and spend a large portion of the day at the same location. The place is chosen using a similar method to choosing the home, independently from other persons. The available POIs in the map are categorized by their type, and weights are assigned based on the category. The weights are part of the configurable settings of the model. The weight for each POI is calculated as

$$W_{\text{interest}} = \frac{A_{\text{polygon}}}{n_{\text{polygon}}} W_{\text{category}} W_{\text{dist}}, \tag{5.5}$$

where A_{polygon} is the area of the bounding box of the polygon that the POI belongs to, n_{polygon} is the number of POIs inside the same polygon and W_{category} is the weight defined for the category. If the POI does not belong to any polygon, default area (A_{default}) will be used. Larger polygons are preferred while the area is divided evenly for each POI that belongs to the polygon.

W_{dist} is the weight factor of the distance. Home of a person is considered to be the anchor point, from which the person travels to different places and thus prefer locations closer to home. It is calculated as

$$W_{\text{dist}}(d) = \begin{cases} (W_{\text{near}} - W_{\text{zero}})\frac{d}{d_{\text{near}}} + W_{\text{zero}} & \text{if } d < d_{\text{near}}, \\ W_{\text{near}}(\frac{d_{\text{near}}}{d})^2 & \text{otherwise,} \end{cases} \tag{5.6}$$

where d is the distance from home to the POI, W_{zero} is the weight at zero distance, and d_{near} is distance where the weight is at W_{near}. The willingness to travel decreases linearly inside the circle defined by the near distance, which matches intuition. For example, if d_{near} was 2 km, W_{zero} was 10 and W_{near} was 1, the probability to visit a close location would be 10 times the probability of a location 2 km away. The value of the constants depend on whether the person is traveling with a car or walking.

After d_{near} the distance weight gradually decays towards zero, so that POIs outside the near distance have at least a small probability to be chosen. When the distances doubles, the weight is decreased to one quarter. The factor never reaches zero, so far away locations are not complete ignored. Figure 5.13 shows the curve of the distance weight.

The exact position to visit inside the office or the school is chosen using a Voronoi

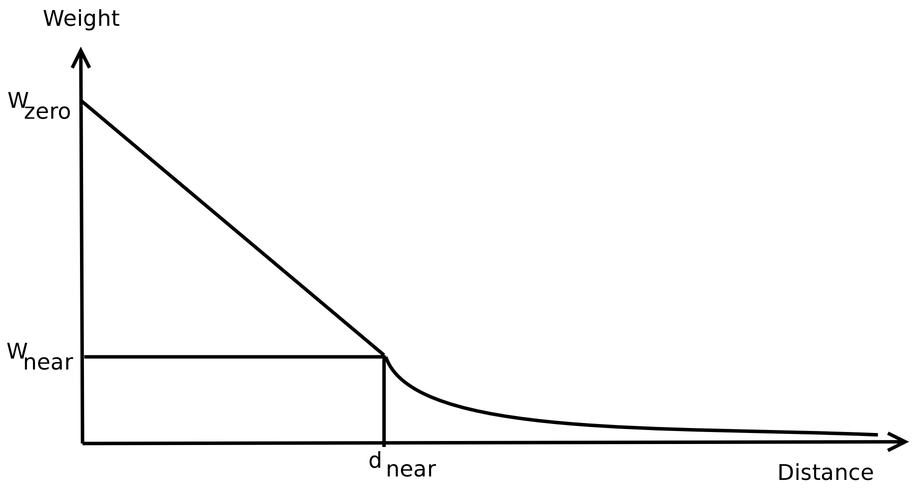


Figure 5.13: The distance weight.

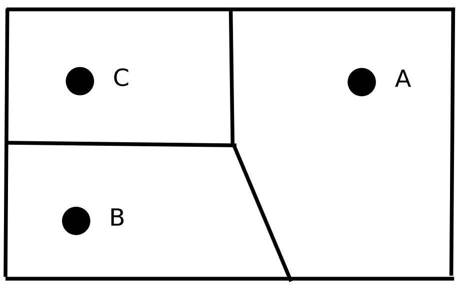


Figure 5.14: An example of a Voronoi diagram of a polygon, used to choose a position inside the polygon.

diagram [2]. The space inside the bounding box of the polygon is divided to cells around each POI. A cell is the area where the distance to other POIs is greater than distance to the POI of the cell. The position to visit is selected from the uniform distribution inside the cell. To avoid constructing a Voronoi diagram, the position can be selected by randomly trying different points until one is found that is inside the correct cell. Figure 5.14 shows an example of a Voronoi diagram of a polygon with three POIs.

5.6.5 Choosing interests

In addition to home and office, each simulated person has a set of interesting places to visit after spending the day at the important location. People tend to have a small number of familiar places to which they like to go. Each person has a predefined number of interests which are selected independent from other persons. For example, each person could have 5–10 interests.

The interests are chosen using a similar method as choosing the office or the school. The POIs are categorized and assigned with configured weights, and the interests are randomly chosen based on the weights. The weights are calculated using (5.6) with the difference that W_{dist} is calculated using the distance from home or the daily place to the POI, depending on which of the distances is smaller. Once people have traveled to the office or the school, the resist to travel to a place further away from home becomes smaller. Also, this increases the probability of visiting interests in the area between home and the office. W_{category} is adjusted to match the willingness to visit an interest. For example, the weight of grocery stores could be increased because people visit the shops daily.

5.6.6 Traveling

The simulated persons move between the locations by traveling along the roads. The traveling happens either by a vehicle or by walking, depending on the distance and whether the person has a car available. The map information contains the type of each road, and the model configuration gives the maximum travel speeds of different road types. The speeds limit depend on whether the person is traveling with a car or walking. Impassable roads are marked by setting the speed limit to zero, for example, to deny using a vehicle on pedestrian paths.

The persons move between two locations using the fastest route, determined with A* pathfinder [17]. The cost function for an edge between two road nodes is the travel time between them, calculated from the speed limit. The path finder must also take into account the one-way roads and separate that can be present in the map information. The path finder produces the same routes as a real navigation device would.

The persons enter and leave the visited locations using a direct path to the nearest road node. The speed to enter and leave is set to very low, because it takes time to enter a large building.

5.6.7 Daily schedule

For each day of the simulation and for each simulated person, a schedule consisting of the visit to the office and to a randomly chosen set of the person's interests is created. The number of interests to visit each day is selected using uniform distribution from a predefined range, e.g. 1–3. The exact positions to visit on each day are chosen randomly using the same method as for selecting the position of the home.

After the interests have been selected, the total distance to travel on the day is determined to choose whether the person walks or uses a vehicle to move between the locations. If the person owns a car, the probability of going on foot on that day is

$$P_{\text{walk}}(d) = \min\left(1, \frac{d_{\text{walk}}}{d}\right), \tag{5.7}$$

where d is the total distance to travel on the day, and d_{walk} is the maximum distance everyone would walk. Outside that distance, the probability of walking halves when the distance to travel is doubled, matching intuition. For example, if d_{walk} is 500 meters, the probability to walk 1 km would be 0.5.

The schedule consists of a sequence of visits with different weights. Leaving from home and coming back are represented as visits to home in the beginning and end of the schedule.

The weight represents the proportion of time the person is willing to spend at the location on that day. The weight of the office or the school is typically larger than the weights of the visits to different interests. In case the weight unit was approximated to be 1 hour, the weight of an interest could be 1 and the weight the office would be 8. Longer visit can be divided to a sequence of shorter visits to model the movement inside a building during the day.

The path finder is used to determine the shortest path and travel time between the positions in the schedule. Then, the travel time is subtracted from the total time available for the day. The remaining time is divided to traveling and different visits based on the weights. The time when the day of a person begins is randomly selected from a uniform distribution to avoid everyone traveling at the same time by increasing the variation of the visiting times. For example, each person could leave between 6:00 and 7:00 in the morning and have total of 16 hours time available, including traveling.

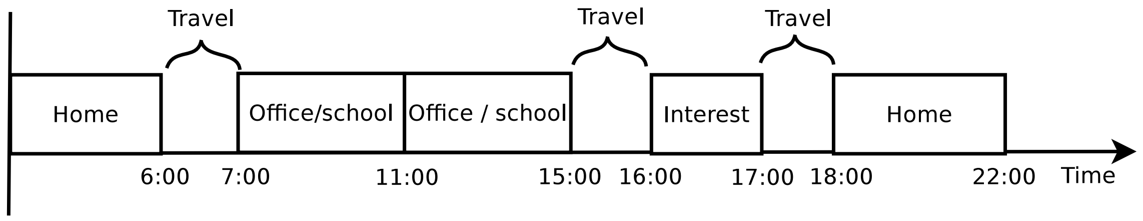


Figure 5.15: An example schedule of a simulated person.

Fig. 5.15 shows an example of a schedule. The person visits two places during the day: The office or the school and one randomly chosen interest. The visit to the office is divided into two visits with different randomly chosen positions, to represent changing the position around lunch time.

6. EVALUATION

This chapter presents the evaluation of Proximate with real user trials and compares the results to the simulator.

6.1 User trial

In the user trial each participant was given an N900 device with the prototype application pre-installed. The members were recruited using a web form asking for basic details of the person and a list of friends. Friends were identified by their email addresses, which were then used to advertise the trial and to find connections between the applicants. The participants were selected from students and staff members who actively spend time in the same buildings at TUT campus and already had some common friends (social networks).

User and technical feedback were continuously collected by automated logs, separate feedback application, online discussion forum, IRC channel and interviews. The applications on the devices were automatically updated and the logs were uploaded to a server when the device was connected to the campus-wide infrastructure WLAN. Figure 6.1 shows how the new versions spread among the participants.

Table 6.1 shows the results of the feedback collected during the trial. The first column is an application feature, and the second is the proportions of 218 participants who selected the property as being one of the best three features of the application.

Over a half othe participants named locality as the most important feature. Of these people, 36% named the radar view and 46% of the participants reported that

Table 6.1: Summary of three best properties in the prototype application.

Property	Proportion of feedback
Locality	54%
Chat	46%
File sharing	29%
Communities	19%
Works everywhere	17%
Easier to meet new people	12%
Zero price communication	9%
Message board	8%
Sharing information with others	7%

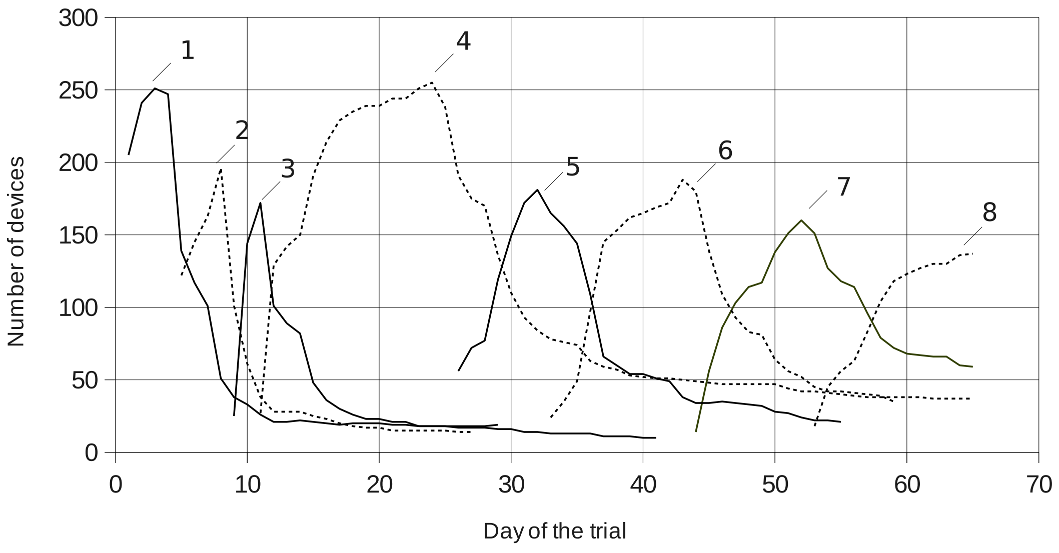


Figure 6.1: Distribution of installed versions of the prototype application on each day of the user trial.

conversations are one of the best properties. 13% of these participants specified that chatting with unknown people is especially important. 12% specified that community wide chats are important. 9% specified that chatting with friends is important. 6% specified that chatting in the same physical event (for example, lecture) is important. 5% specified that chatting in private is most important.

The private and the community conversation are the most important application features in Proximate. The lack of reliability in networking protocols caused 28% of users to complain in feedback, especially about chat.

6.1.1 Technical evaluation

The devices logged information about encounters during the user trial. Figure 6.2 presents the number of times a user has encounter another user each day of the user trial. The trial started on Monday, and the weekends are clearly visible as less active periods. In the beginning, some of the users encountered almost all other devices. The startup meeting held on the 8th day is also clearly visible as a spike. On that day, each user encountered on average half of the other participants. The long weekend on the third week caused the first longer inactive period, after which the activity stabilized to a lower level for the rest of the trial.

The logs were used to analyze DTN in local social networking. The reach of information from a single device can be extended by delay-tolerant communication, where other devices participate in carrying and replicating messages between users that have no direct contact. The analysis was done with a simulation reusing the trial

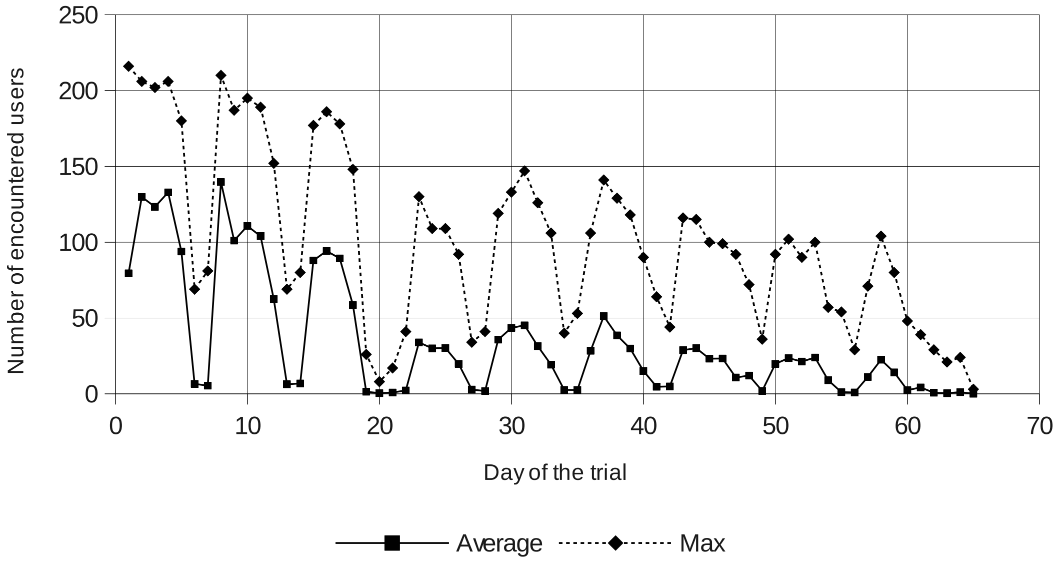


Figure 6.2: The number of other users each user has encountered on each day of the user trial, on average and at maximum. The highest possible count is 249.

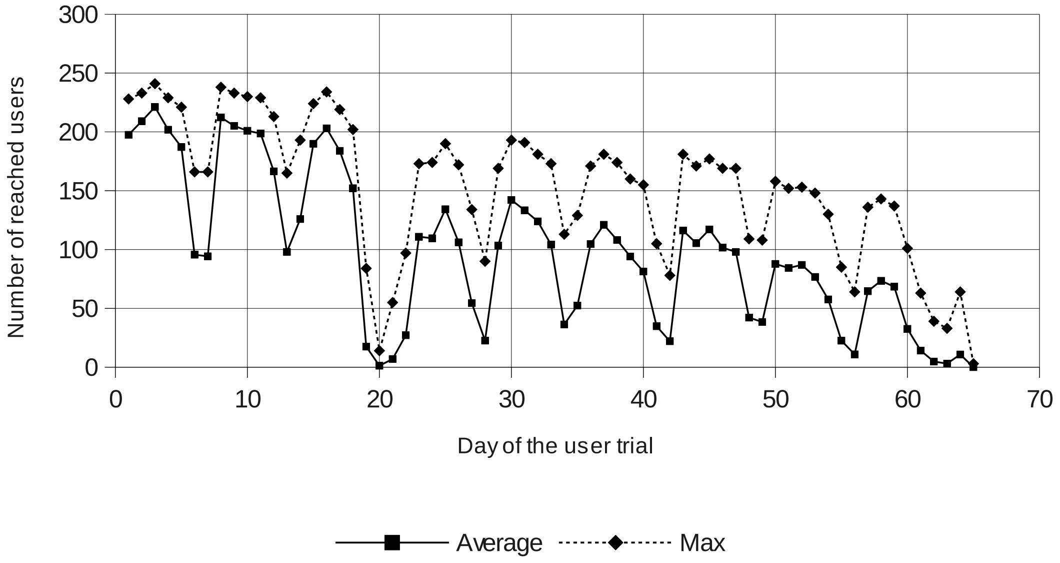


Figure 6.3: The average number of indirectly reached users on each day of the user trial, on average and the maximum. The highest possible count is 249.

Table 6.2: Distribution of different actions an average user performed during the user trial.

Feature	Number of times used
Send a conversation message	88.6
Download a shared file	4.6
Create a new share (incl. messages)	2.9
Publish a new message	1.3
Direct upload of a file	0.4

data. It was assumed that the devices exchanged every message on each encounter.

Figure 6.3 shows the number of indirectly reached users on each day of the user trial. The number tells how many other users would receive a message during a 24-hour period after being published. The figure indicates that DTN improves the reach of information by having on average twice the number of reached users when compared to direct communication. The average reach is also almost as high as the maximum reach during the first three weeks when almost every device were reachable.

Table 6.2 presents the number of times different actions were performed during the user trial. The first column gives the feature, and the second the number of times a user invoked the feature. As the chat was implemented in a plugin independent from the share system, the number is presented separately. Publishing a message implies creating a new share, and the number of created shares includes published messages.

The prototype had the ability to directly upload files to other users, without first sharing the file. The numbers do not include unsuccessful attempts, and many of the actions failed due to the wireless multi-hop communication. The upload and download numbers also include attempts that failed in the middle of the transfers. The most commonly used feature was sending a message, and an average participant sent almost one hundred messages using Proximate during the trial.

Table 6.3 shows the different file types that were transferred, identified by the file extension. The first and the second columns represent the file extension and a description, the third column gives the number of times a shared file was downloaded and the fourth column shows the number of times a file was directly uploaded to another user. Sharing copyrighted content, such as commercial music and videos, was denied during the trial.

The most often transferred files were pictures in JPEG format, such as the photos taken built-in camera in N900. The second most common file type was mp3, which is the most common format for storing digital music. MPEG-4 files were also transferred often, most likely because it is the format in which videos taken with the N900 are stored. People also used the prototype to exchange software packages

Table 6.3: Distribution of file name extensions that were downloaded and uploaded during the trial.

Suffix	Description	Downloads	Direct uploads
jpg/jpeg	Picture (from a camera)	540	46
mp3	Music and audio	208	19
mp4	Video	167	9
aac	Music and audio	102	1
png	Image and graphics	44	7
deb	Software package for N900	37	2
3gp	Video taken with N900	22	0
pdf	PDF-document	9	1
nes	Nintendo ROM image	7	2
avi	Video	7	0
html	Web page	5	0

and ROM images for a Nintendo simulator between the devices.

6.1.2 What was learned

Many of the participants complained about reliability of communication during the user trial. Network packets were sometimes lost, which caused chat messages not to reach other people, which participants found annoying in private chats. The users often had to repeat what they said to get the message over.

The underlying WLAN mesh network did perform retranmissions, but it was found that they were not sufficient when the devices became unreachable for a longer periods of time. The share system was improved by adding acknowledges, and distributed a new version during the trial. An icon was included next to a chat message in the user interface to indicate state of delivery.

Another problem with is that information can disappear at any time. When the user wants to download an interesting shared file, it might be already gone from the network. This can be solved by automatically downloading contents when it becomes available. The user could subscribe to content in advance by entering a set of keywords of interest. When a shared file matching a keyword is found, it would be automatically downloaded and the user would be notified.

Subscriptions depend on the user knowing the exact keywords that might be interesting before hand. This is against the requirement of serendipity, where the application can suggest new interesting content that the user was previously unaware of. A better alternative would be use learning methods to evaluate how interesting available content is. The user could direct the learning by giving feedback on suggested content.

Table 6.4: Mobility model parameters

Parameter	Symbol	Value
Weight at zero distance	W_{zero}	10
Weight at near distance	W_{near}	1
Near distance, vehicle	d_{near}	10 km
Near distance, on foot	d_{near}	2 km
Walking distance	d_{walk}	1 km
Default area	A_{default}	36m^2
Enter/leave speed	-	1.0 m/s
Number of interests	-	10
Interests to visit	-	1 – 4
Number of nodes	n	250
Day length	-	16 hours
Day begins	-	6:00 – 7:00
Number of days	-	5
Nodes with a car	-	50%

6.2 Simulations

The behavior of the mobility model was compared to a user trial. Data collected from the pilot was analyzed to determine which devices were connected at any particular time. A period between the midnights of Monday and Friday from the third week of the trial was used for comparison.

The mobility model was configured to use the map of Tampere from OpenStreetMap. The model was adjusted to match the participants of the user trial. The number of nodes to simulate was set to 250, and length of the simulation was 5 days. The weight of the interest presenting the campus was increased so that the nodes would visit the campus often. The campus is located 10 kilometers from the city center, and while many participants reside near the university, others commute daily from other parts of Tampere. It was assumed that 50% of the participants owned a car. Table 6.4 shows the simulation parameters.

A simple simulation tool was created to record the encounters from the mobility patterns. The simulator plays back the movement according to the given paths and simulates peer discovery between the devices. The simulation happens in 10 second intervals, and every device sends a single discovery packet on each step. The devices that receive the packet are considered to be visible. Multi-hop networking or DTN were not included in the simulation.

The simulation tool applies a simple packet loss model, which is modeled after IEEE 802.11 WLAN with at the lowest data rate of 1 Mbps [36]. The length of the discovery packet is 500 bits. Table 6.5 lists the radio parameters.

Table 6.5: Radio parameters

Parameter	Symbol	Value
Transmitter power	P_t	15 dBm
Signal loss	L_c	-20 dB
Background noise	N_0	-84.4 dBm/MHz
Frequency	f	2.4 GHz
Bit rate	R_b	1 MHz
Bandwidth	B	20 MHz
Packet length	l	500 bits

6.3 Analysis

The encouters from the user trial were compared to the data gathared from the discovery packets of the simulation. The interval between subsequent encounters and the duration of the encounters gives information on how strongly the nodes are connected [25]. The distribution of intervals indicates how often two nodes can exchange information. The more often the nodes encounter, the faster new information will spread which can be important in social applications. There can be a set of nodes which encounter often and a long tail of nodes that may encounter just a couple of times. Especially in DTN, the flow of information depends on the encouner patterns of all nodes.

The duration of the encounters defines how the information is exchanged. Short encounters require fast synchronization of information while longer encounters allow session-based communication, such as chatting and browsing shared content. Longer encounters are also needed to transfer large files.

Figure 6.4 shows the cumulative distribution of time between subsequent encounters of node pairs. A node pair is an unordered combination two different nodes. Thus, the number of node pairs for $n = 250$ nodes is $n(n-1)/2 = 31{,}125$.

The distributions from user trial and the simulation follow the same shape. The proportion of encounters that have interval of over 1.16 days (100,000 s) have the same distribution, but below that the intervals are longer in the simulation than in the trial.

Figure 6.5 shows the cumulative distribution of the duration of encounters between node pairs. In case a node has multiple encounters active at the same time, each are counted separately.

The distribution is the same for the user trial and simulation for durations in range of less than 16 minutes (1,000 s). For longer durations, the simulation tends to produce longer encounters than were observed in the user trial. The longest encounter in the simulation and in the trial was 14 hours.

Figure 6.6 shows the distribution of number of other nodes a node has seen

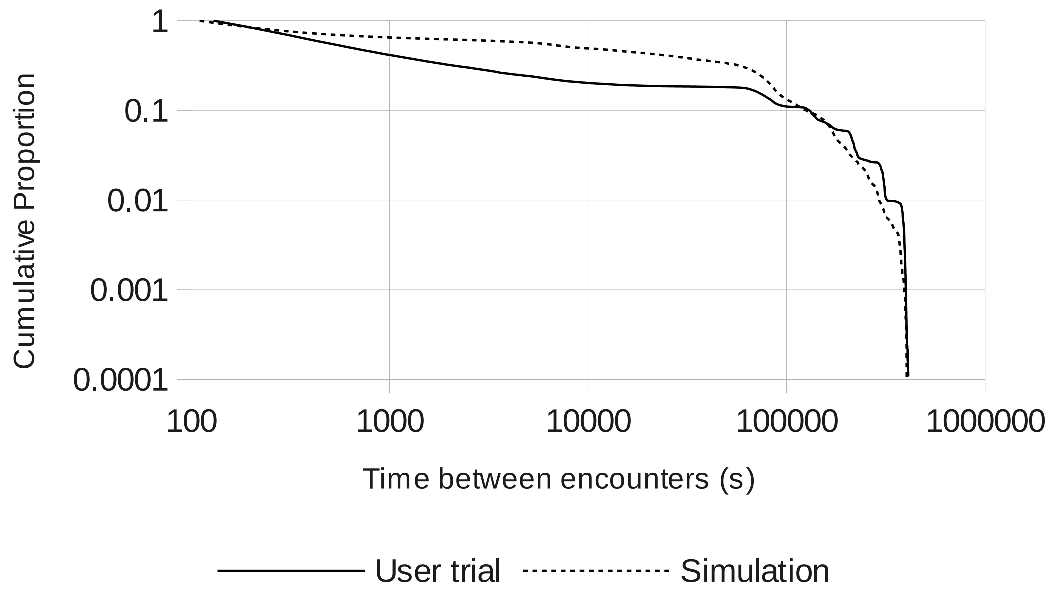


Figure 6.4: Cumulative distribution of time between subsequent encounters of node pairs. The horizontal axis gives an interval in seconds, and the vertical axis gives the proportion of encounters which were longer apart than the interval.

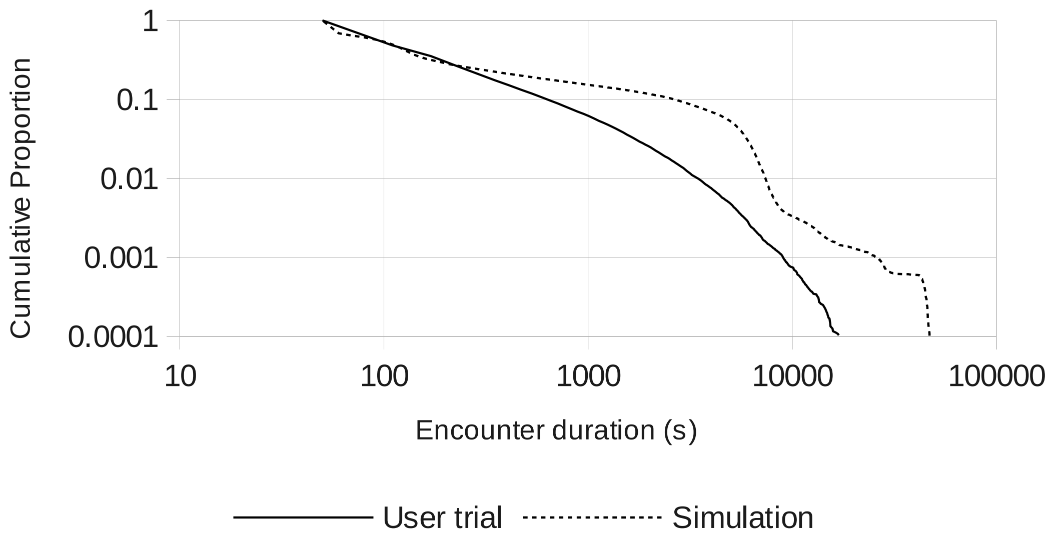


Figure 6.5: Cumulative distribution of the duration of encounters between node pairs. The horizontal axis gives an encounter duration in seconds, and the vertical axis gives the proportion of encounters that were longer than the duration.

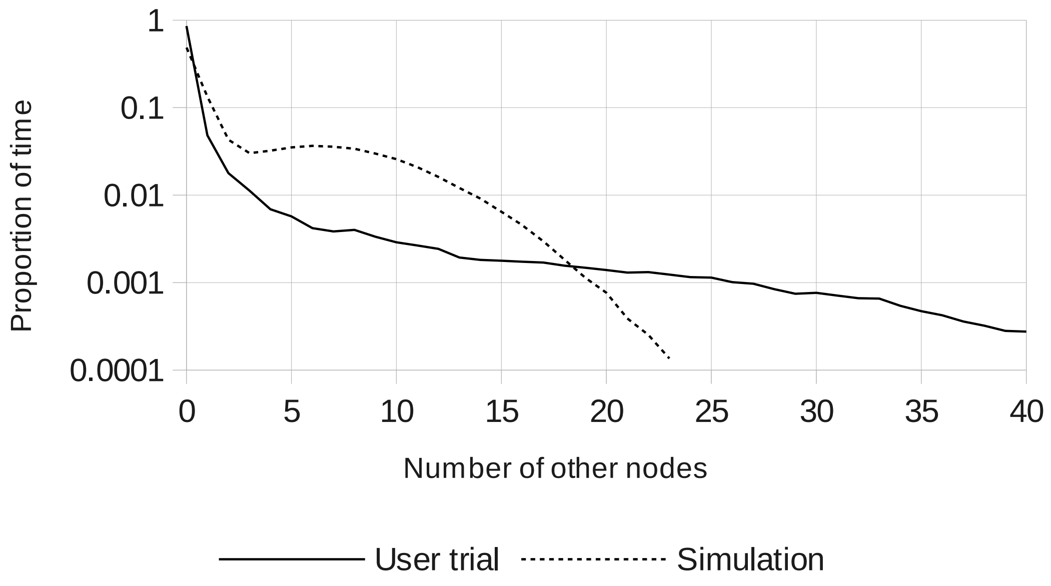


Figure 6.6: Time distribution of encountered nodes. The horizontal axis gives a number nodes, and the vertical axis gives the proportion of time that many nodes were in proximity.

on average. Most of the time the nodes were alone, which is the highest spike in the graph. Nodes in the simulation tend to see more nodes in proximity and the maximum number of nodes seen was smaller than in the trial. This may be explained by the use of multi-hop routing in the user trial, which increases the number of reachable devices in the presence of a few devices that do routing.

7. CONCLUSION

7.1 Social application

This thesis defines the requirements and use cases for social applications that use local communication. Because of the opportunistic nature, the application should be always active, automatically discovering new contents with small impact on battery life. This can be accomplished by minimizing the required network traffic. The application should implement cryptography to secure the users' private information. To account for future popularity, the application should be scalable and allow the communication protocols to be extended.

This thesis also proposed a design to implement the use cases. The design does not require any particular communication network but can benefit for having power saving mode that increments the battery life at the cost of increased latency and smaller throughput. When required, a higher throughput can be obtained by switching to ad hoc WLAN or similar between the devices.

Sharing of information is unified into a single share architecture, where the same data format is used for different types of shares. A share has an optional text content and can have files attached. The shares can be linked to form continuous conversation chains and small shares can be automatically replicated to increase the reach to a larger area using DTN.

Findings from the trial reports local conversations as the most important use case of the application. The trial analysis show that DTN almost doubles the average reach of published information.

Future research topics were raised by the design social application and user trial. First, we need a method for automatically learning user preferences to screen and suggest content. Second, we need to develop a distributed database for optimizing the traffic load when replicating content. Third, new use cases and user trials should be developed.

7.2 Simulator

The simulation model presented in the thesis can be used to quickly evaluate prototype applications that use direct communication between mobile devices. The model gives more accurate results over previously used random models because it simulates the behavior of average people commuting between home, office and interests either by walking or using a vehicle. The model uses weighted random selection to match the intuition of how people are likely to behave. The model was compared against mobility behaviour gathered from a user trial, where it was found out that the model matches the average person living in an urban environment.

Another advantage is that the configuration of the model happens a high level instead of requiring each simulation person to be configured individually. This allows comprehensive testing in multiple different scenarios can be carried out by adjusting the parameters.

In the future, the model could include the social network between persons and use the relationships to increase the the probability of people visiting the same place at the same time. People often use public transportation to travel in urban environments, and the path finding can be made more accurate by including real time tables and routes of public transportation.

BIBLIOGRAPHY

[1] A. Ahtiainen, K. Kalliojarvi, M. Kasslin, K. Leppanen, A. Richter, P. Ruuska, and C. Wijting. Awareness networking in wireless environments. *Vehicular Technology Magazine, IEEE*, 4(3):48 –54, September 2009.

[2] Franz Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, September 1991.

[3] A. Beach, M. Gartrell, S. Akkala, J. Elston, J. Kelley, K. Nishimoto, B. Ray, S. Razgulin, K. Sundaresan, B. Surendar, M. Terada, and R. Han. WhozThat? evolving an ecosystem for context-aware mobile social networks. *Network, IEEE*, 22(4):50–55, July 2008.

[4] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. PeerSoN: P2P social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, SNS '09, pages 46–52, New York, NY, USA, 2009. ACM.

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

[6] Bram Cohen. The BitTorrent protocol specification. `http://www.bittorrent.org/beps/bep_0003.html`, June 2009.

[7] OpenStreetMap contributors. OpenStreetMap. `http://www.openstreetmap.org/`, 2012.

[8] P. Costa, C. Mascolo, M. Musolesi, and G.P. Picco. Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. *Selected Areas in Communications, IEEE Journal on*, 26(5):748 –760, June 2008.

[9] Douglas Crockford. RFC 4627 – The application/json media type for JavaScript Object Notation (JSON). Technical report, July 2006.

[10] Nathan Eagle and Alex Pentland. Social serendipity: Mobilizing social software. *IEEE Pervasive Computing*, 4:28–34, April 2005.

[11] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. Working day movement model. In *Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models*, MobilityModels '08, pages 33–40, New York, NY, USA, 2008. ACM.

[12] M. Esbjörnsson, O. Juhlin, and M. Östergren. The hocman prototype: Fast motor bikers and ad hoc networks. In *Proceedings of MUM, Oulu, Finland*, pages 91–98, 2002.

[13] Facebook, Inc. Facebook timeline. `http://www.facebook.com/press/info.php?timeline`, 2012.

[14] Facebook, Inc. Share where you are. `http://www.facebook.com/about/location`, 2012.

[15] Google. Google+: real life sharing rethrought for the web. `https://plus.google.com`.

[16] M. Gunes, F. Juraschek, B. Blywis, and C. Graff. MoNoTrac – a mobility trace generator based on openstreetmap geo-data. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 618 –623, November 2010.

[17] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100 –107, July 1968.

[18] James O. Henriksen, Robert M. O'Keefe, C. Dennis Pegden, Robert G. Sargent, and Brian W. Unger. Implementations of time (panel). In Douglas W. Jones, editor, *Proceedings of the 18th conference on Winter simulation*, WSC '86, pages 409–416, New York, NY, USA, 1986. ACM.

[19] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, pages 241–250, New York, NY, USA, 2008. ACM.

[20] D.N. Kalofonos, Z. Antoniou, F.D. Reynolds, M. Van-Kleek, J. Strauss, and P. Wisner. MyNet: A platform for secure P2P personal and social networking services. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 135–146, March 2008.

[21] Steffen Kern, Peter Braun, and Wilhelm Rossak. MobiSoft: An agent-based middleware for social-mobile applications. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4277 of *Lecture Notes in Computer Science*, pages 984–993. Springer Berlin / Heidelberg, 2006.

[22] N.D. Lane, E. Miluzzo, Hong Lu, D. Peebles, T. Choudhury, and A.T. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, September 2010.

[23] Yao-Nan Lien, Hung-Chin Jang, and Tzu-Chieh Tsai. Design of P2Pnet: An autonomous P2P ad-hoc group communication system. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM '09. Tenth International Conference on*, pages 649–654, May 2009.

[24] A. Mei, G. Morabito, P. Santi, and J. Stefa. Social-aware stateless forwarding in pocket switched networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 251 –255, April 2011.

[25] Mirco Musolesi and Cecilia Mascolo. A community based mobility model for ad hoc network research. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, REALMAN '06, pages 31–38, New York, NY, USA, 2006. ACM.

[26] National Institute of Standards and Technology. *FIPS PUB 180-1: Secure Hash Standard*. Gaithersburg, MD, USA, 1995.

[27] National Institute of Standards and Technology. *FIPS PUB 800-57: Recommendation for Key Management – Part 1: General (Revision 3)*. Gaithersburg, MD, USA, July 2012.

[28] Tom Nicolai, Eiko Yoneki, Nils Behrens, and Holger Kenn. Exploring social context with the Wireless Rope. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4277 of *Lecture Notes in Computer Science*, chapter 112, pages 874–883. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[29] Nokia Conversations. Nokia Instant Community gets you social. `http://conversations.nokia.com/2010/05/25/nokia-instant-community-gets-you-social/`, 2010.

[30] Nokia Corporation. Home of the Maemo community. `http://maemo.org/`, 2010.

[31] OpenStreetMap contributors. OpenStreetMap Statistics. `http://www.openstreetmap.org/stats/data_stats.html`, 2012.

[32] A-K Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. Mobiclique: Middleware for mobile social networking. In *WOSN'09: Proceedings of ACM SIGCOMM Workshop on Online Social Networks*, August 2009.

[33] J. Porras, P. Hiirsalmi, and A. Valtaoja. Peer-to-peer communication approach for a mobile environment. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, page 7 pp., January 2004.

[34] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21:120–126, February 1978.

[35] Bluetooth SIG. *Bluetooth Specification Version 2.0.* Bluetooth SIG, November 2004.

[36] IEEE Computer Society. IEEE standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, December 2007.

[37] Michael Terry, Elizabeth D. Mynatt, Kathy Ryall, and Darren Leigh. Social net: using patterns of physical proximity over time to infer shared interests. In *CHI '02 extended abstracts on Human factors in computing systems*, CHI EA '02, pages 816–817, New York, NY, USA, 2002. ACM.

[38] The GNOME Project and PyGTK Team. PyGTK. `http://www.pygtk.org/`.

[39] Elaine Toms. Serendipitous information retrieval. In *In Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries*, pages 11–12, 2000.

[40] Ian Vo, T. J. Purtell, Ben Dodson, Aemon Cannon, and Monica S. Lam. Musubi: A mobile privacy-honoring social network. `http://mobisocial.stanford.edu/papers/musubi.pdf`, September 2011.

[41] Kaisa Väänänen-Vainio-Mattila, Petri Saarinen, Minna Wäljas, Marko Hännikäinen, Heikki Orsila, and Niko Kiukkonen. User experience of social ad hoc networking: findings from a large-scale field trial of twin. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, MUM '10, pages 10:1–10:10, New York, NY, USA, December 2010. ACM.

[42] Hayoung Yoon and Jongwon Kim. Collaborative streaming-based media content sharing in WiFi-enabled home networks. *Consumer Electronics, IEEE Transactions on*, 56(4):2193–2200, November 2010.