Improving Node Discovery in Delay Tolerant Networks by Exploiting Location History

A Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science (M.Sc.) in Applied Computer Science

at the

Berlin University of Applied Sciences (HTW)

First Supervisor: Prof. Dr. Jürgen Sieck

Second Supervisor: Prof. Volodymyr Brovkov

Submitted by B.Sc. Mathias Lenz

14. September 2012

Contents

Kurzbeschreibung

Die Suche nach Nachbarknoten, auch Discovery genannt, ist eine Grundvoraussetzung für die Kommunikation in mobilen Ad-hoc Netzwerken (MANET). Eine spezielle Art von MANET sind die "Pocket Switched Networks", Netzwerke die zwischen Geräten aufgebaut werden, die Menschen bei sich tragen. Diese Netzwerke können dafür genutzt werden, die steigende Last auf Infrastrukturnetzwerken wie GSM oder UMTS abzufangen. In dieser Masterarbeit wird ein adaptives Discoveryverfahren vorgestellt, dass menschliche Mobilität aufzeichnet und ausnutzt, um die Suche nach Geräten zu verbessern. Das Verfahren basiert auf der Identifizierung von relevanten Orten und der Vorhersage zukünftiger Mobilität. Durch Kontextinformationen des Ortes wird der Intervall, in dem die Discovery gestartet wird, dynamisch angepasst, um mit möglichst wenig Stromverbrauch eine große Anzahl an Geräten zu finden.

Abstract

Discovering neighboring nodes is a prerequisite for data transfer in mobile ad-hoc networks (MANETs). "Pocket Switched Networks" are a special kind of MANET, in which the participating devices are carried by humans. Pocket Switched Networks can be used to lessen the strain on infrastructure networks like GSM or UMTS. This thesis will describe an adaptive discovery scheme, which logs and uses mobility data in order to improve the discovery. The scheme is based on identifying places of interest and predicting future mobility. Using contextual information of the current location or path, the discovery interval can be adapted dynamically to discover as many neighboring nodes as possible, but with limited energy consumption.

1 Introduction

1.1 Discovery in Delay Tolerant Networks

"Delay Tolerant Networks" (DTN) are a special kind of network where nodes may not be connected at any given time. Routes may have a very long delay and the network might partition regularly. Thus, these networks work according to the "Store and Forward"-principle, meaning that they may temporarily persist network packages to send them to further nodes later. This stands in contrast to "traditional" protocols like TCP/IP. DTNs are a fairly recent research topic, mostly based on previous research on ad-hoc networks. Environments where DTNs may be used include mobile networks, interplanetary or deep-space networks, military ad-hoc networks or sensor networks[Fal03].

Special subtopics of delay tolerant networking have been defined, mostly based on several characteristics of the network. "Pocket switched networks"[HCS+05] has been used as a special term for opportunistic networks between users of small mobile appliances, allowing them to send and receive data in environments without infrastructure, bridging the way into larger networks through local wireless technologies like Bluetooth or WiFi. "Participatory networks"[BEHP06] act similarly in that they try to incorporate the users mobile devices. Participatory sensing is a special case in which mobile devices gather sensor information, either using their own built in sensors or receiving them from nearby sensor systems.

The popularity of modern smartphones makes participatory systems especially interesting, because these devices often include sophisticated sensor systems, high processing capabilities and network connectivity via Bluetooth, WiFi and mobile broadband. These devices are able to replace specialised hardware installations in certain situations.

One especially useful feature of most smartphones is their location awareness. Some routing protocols in opportunistic networks are able to exploit this information to increase performance, e.g. by calculating the best route depending on proximity to neighboring nodes. This idea is already used in certain sensor networks which forward

packets to nodes who can serve the most additional area[WC02]. Several other protocols used in delay tolerant networks use encounter information, forwarding to nodes with a higher probability of coming in contact with the destination node. More recent routing protocols even try to use social information of the networks participants, identifying peer groups and popular peers[HCY10].

There are common problems in participatory networks and traditional sensor networks. The most prevalent is energy efficiency, based on the fact that mobile devices run on a battery. Most network connections are expensive in terms of energy consumption, making it necessary to let them run as briefly as possible. Another problem in opportunistic networks is neighbor node discovery. Knowledge of nodes in proximity is essential before initiating any kind of data transfer. Bluetooth and WiFi include discovery mechanisms which are costly in terms of processing and battery power, they should therefore be issued as rarely as possible.

This thesis proposes a dynamic discovery of neighboring nodes, based on location information stored in the mobile devices. Location information, along with additional routing-related data like encounters or time of day can be used to initiate neighbor discovery only when it is deemed useful. The interval and duration of discovery can be adjusted based on the current nodes' location and the location of nodes it has encountered. Along with the dynamic discovery, an application architecture will be described which is able to run dynamic discovery mechanisms and provide mobility and contact data to other applications.

The system will run on an Android smartphone, using Bluetooth for neighbor discovery. Evaluation will be done based on logging data captured over a period of about 3 months.

1.2 Overview of this thesis

The second chapter will provide an introduction to the general research space of Delay Tolerant Networking. It will provide information on human mobility and its significance for a special kind of DTN: Pocket Switched Networks. Human mobility will be described in terms of metrics relevant to these networks. The following sections describe ways to model and predict mobility, which will be the basis of the discovery scheme proposed in this paper. Afterwards, some algorithms related to discovery and mobility will be described. The chapter ends with some general information on discovery using Bluetooth, which is the technology used to evaluate the proposed scheme.

The third chapter describes the approach the discovery scheme will use to exploit contact patterns at specific locations. Here, the relevancy of some of the metrics described in chapter 2 will be assessed. To filter relevant locations, a spatial clustering algorithm is described. The clustered locations are the basis for a mobility prediction algorithm described afterwards. Finally, the approach to adapting the discovery to the location history is described.

The fourth chapter will describe the application used to capture contact and location data. The description includes the data model and the application architecture. The chapter ends with a description of the discovery scheme implementation.

In chapter five, the algorithms described in the third chapter will be evaluated, based on the data captured by the application outlined in chapter four. The evaluation includes the clustering and mobility prediction, and the discovery scheme will be compared to the scheme it is based on.

At last, the insights gained in the development of the scheme will be discussed.

2 Discovery in Delay Tolerant Networks

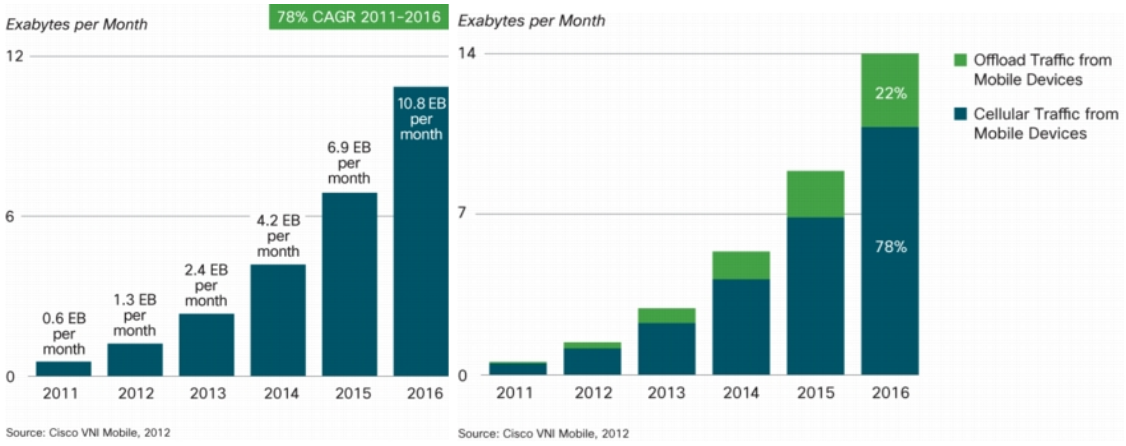
Neighbor node discovery is a requirement for data exchange in highly mobile networks. Improving discovery times leads to more throughput in the network, but the discovery process itself is often a strain on other resources like battery power. Adapting to properties of the network is one way to decrease power consumption. The following chapter will consider discovery in delay tolerant networks, with a focus on networks driven by human mobility. These networks are established mostly between devices carried by users, though some of them may provide network links to infrastructure networks. The first part of this chapter will identify some of these networks and their use within the larger research space of delay tolerant networking. The next part focuses on human mobility and its impact on networking. The following parts will provide information about adaptive discovery algorithms and the discovery process in Bluetooth.

2.1 Delay Tolerant Networks

Delay Tolerant Networking was first described in [Fal03] as a means to bring intra- and inter-network connectivity to areas where typical network protocols and architectures like TCP/IP are not appropriate. Networks in these areas may experience long delays or faulty connections, and participating devices may have very limited power or computing resources. The idea of connecting these networks with more reliable networks like the Internet led to an architecture that incorporated the idea of optional reliability and asynchronicity. DTNs were envisioned for a large number of uses, including near-Earth satellite communications, deep space communications, terrestrial mobile networks or sensor networks.

Delay Tolerant Networks operate on the "Store and Forward"-principle by storing data on the local device and opportunistically transmitting it to neighboring nodes based on some forwarding scheme.

Pocket Switched Networks are considered a part of the Delay Tolerant Network research space [HCS+05]. They are used to transfer data between the mobile devices of



(a) Mobile Data Traffic (b) Offload Traffic

Figure 2.1: Forecasts of mobile data traffic and offload [Cis]

users in an opportunistic way. PSNs were first proposed as a way to enable network connectivity between "connectivity islands", such as WiFi at home or at work. While commuting between these locations, data transfer may be slow or expensive or may not be available at all. Unused processing power and network bandwidth, for example via WiFi or Bluetooth, may be used to establish networks without an external infrastructure. Hui, et al "envision a world where these resources can be used to provide networking functionality alongside access networks, and where users' applications make use of both types of bandwidth transparently." [HCS+05]

The target devices of PSNs are those carried by users which are active all or at least most of the time. These devices include mobile phones, PDAs, Laptops and, more recently, Smartphones. The surge in smartphones sales is expected to continue, impacting the current mobile data infrastructure. The number of mobile phones is expected to surpass the worlds population during 2012 and global mobile network traffic will increase 18-fold between 2011 and 2016[Cis]. The increasing number of mobile phones also results in more density, which can be exploited by deploying infrastructure-less networks. Pocket Switched Networks may therefore be able to ease the strain on the infrastructure by offloading traffic or disseminating data to nearby nodes. Most smartphones switch to a available WiFi network when the opportunity arises, which already offloads much of the data traffic produced by mobile and portable devices.

In [MW11] the authors use a hybrid routing system to integrate networks with and without infrastructure. The routing is determined on a per-message basis, with the local delay tolerant network being the preferrable data carrier. Messages are routed to a

single recipient on the idea of "assurance of successful delivery through infrastructure". DTN routing is performed as long as its delivery is deemed likely within the time-to-live (TTL) of the message. If the TTL is nearly elapsed, the infrastructure is used for delivery. In the case of a network where all devices are capable of using ad-hoc communications as well as the infrastructure, the system is able to offload 36% of traffic under a TTL of 5 hours.

[WAL11] discusses a framework for data dissemination that guarantees short delivery delays. The system uses acknowledgements to determine if new copies the data to be disseminated should be injected into the opportunistic network. Content is propagated epidemically from a subset of nodes that keep track of delivered content to determine if additional copies need to be injected into the network.

Han, et al [HHKM10] also propose a system for data dissemination. Here, a content provider chooses a set of users to which it sends data. These users deliver the data to other subscribers through opportunistic networking. If a user does not receive the data in a certain amount of time, the service provider will deliver it directly through the infrastructure.

2.2 Human Mobility

This section will provide some insights on human mobility in mobile networks. First, several properties of human mobility in regards to networking contacts will be described. These properties were identified in several studies which analyzed real-world mobility data in order to provide researchers a number of metrics relevant to ad-hoc networking. Afterwards, research in the field of mobility modeling and mobility prediction will be presented. While mobility modelling is especially useful in simulations, it is also the basis of some mobility prediction algorithms.

2.2.1 Properties of Human Mobility in Mobile Networks

There are a several metrics that are of interest when talking about discovery in wireless ad-hoc networks with mobile users. The following section will provide an overview of some, based on a study done by the National University of Singapore[NM07]. This study used 12 devices, of which 9 were mobile and 3 were stationary, to probe for neighboring nodes using the Bluetooth protocol. The mobile nodes were used by students and faculty members of the university. The devices conducted a bluetooth dis-

covery process every 30 seconds. Logging was done for four months, producing about 350.000 log events and discovering over 10.000 unique devices. A previous study by Chaintreau, et al[CHC05] only referred to inter-contact and contact time. It analyzed 4 different data sets, two for bluetooth and two for WiFi traces.

In conclusion, the researchers found that several metrics approximately follow a power law. Plots for a selection of metrics can be seen on 2.2 on page 13. The analysed metrics are as follows:

Contact time is the duration in which two devices are in contact, meaning that they are in communication range. Longer contact times lead to more potential throughput and therefore more data exchanged during each contact. The analysed dataset revealed that contact time is one of the metrics following a power law, where "80% of the contacts are short in duration lasting less than 9 minutes."[NM07] Contact times are also independent of user behavior, with the mean slope of distribution having a variance of 0:02 across mobile users. Other experiments also found the contact time to follow a power law, although with varying decay coefficients [CHC05, MV03, SCP04].

Inter-pair-contact time describes the time between two devices being successively in contact with each other. If a pair of devices is only in contact once, their inter-pair-contact time is infinite. Ths metric is useful for DTN applications because it potentially influences the choice of a forwarding algorithm "to maximize the successfull transmission of messages in a bounded delay." [CHC05] Based on the datasets acquired in the aforementioned study, this metric does not seem to follow any power law. The slopes for different users were also fairly different. The authors speculate that this may be due to individual behaviour of the participants. Nevertheless, it also found that 80% of inter-pair contacts occur within two hours. Some differences between Bluetooth and WiFi traces were suspected to be due to the more mobile nature of the Bluetooth devices, which were carried all the time. This also indicates an opportunity for highly mobile devices like smartphones.

Inter-contact time is the time between two successive contacts for a single device to any other device. This metric also follows a power law, with 80% of inter-contact times occuring within 40 minutes. Furthermore, most inter-contact times were between 4 and 13 minutes when constraining the data to workdays between 8AM and 8PM. This metric determines the frequency in which a mobile node can exchange data, and may also be useful to infer the penetration of appropiate mobile devices.

Meeting time identifies the time when a device is continuously in contact with at least one other device. Meeting time depicts the aggragation of groups. It is also approximately distributed according to a power law, with 80% of meetings lasting less than 30 minutes. Understanding the way people aggregate in groups can be inferred from this metric.

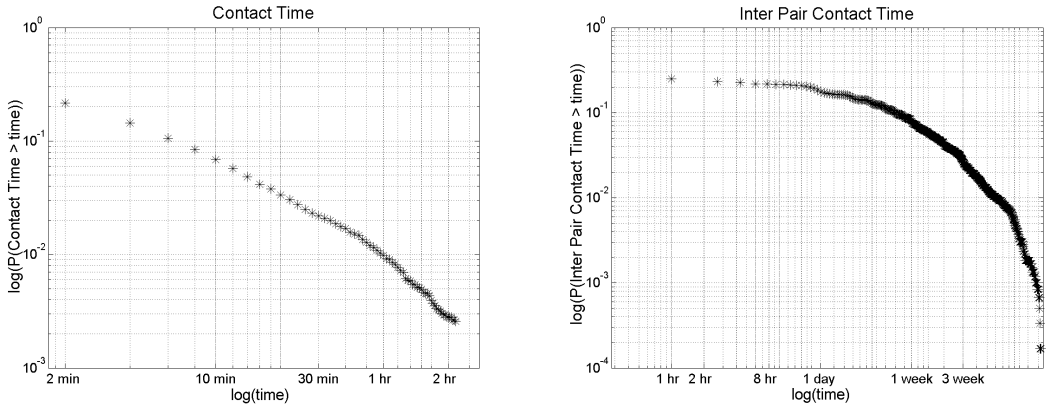
Inter-meeting time is the interval between two meetings for a given device. This is also approximated by a power law and is similar to the distribution of inter-pair contact times.

Meeting size describes the number of devices that are in contact during a meeting. The study found that meeting time correlates to meeting size, with meetings of size 1 (2 devices) lasting 13 minutes on average and meetings of size 2 and above lasting 17 minutes on average. 80% of meetings were of size 1, while 80% of contacts were made in meetings of at least size 2, implying that aggregation centres are potentially useful for DTN applications.

Average Instantaneous Meeting Size is computed for every meeting "by weighting each device in the meeting by the fraction of time for which that device was in contact with the probe." [NM07] A low value identifies a dynamic environment, while values near the meeting size depict a fairly static meeting. Most meetings had this value at about 1, meaning that users are mostly discovered in dynamic environments. This implies that data exchange should be immediate, because every device may leave the meeting quickly.

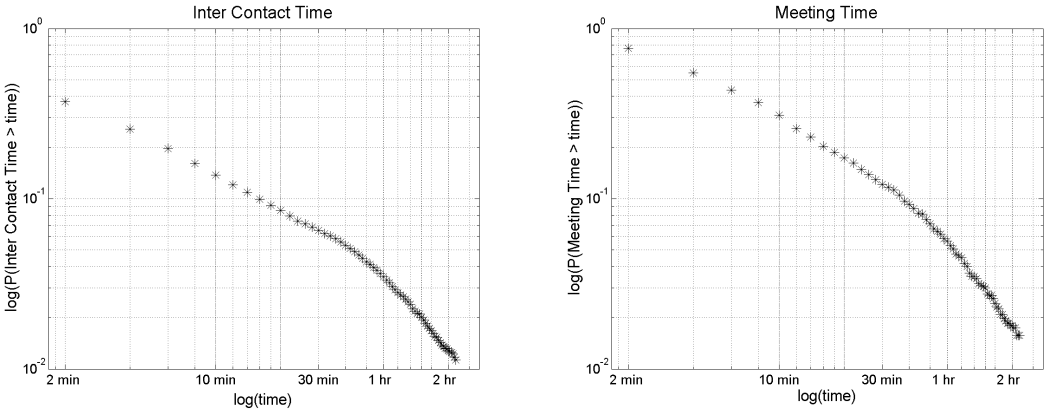
The study also tried to find patterns in the mobility based on the time of contacts. In particular, the researchers found that most participants had a clear diurnal schedule, with no devices discovered between midnight and 8AM. Apart from this, the variance of discovered devices was large during other times of the day. When analysing the number of contacts made on each day, there was also a large variance. This indicates that estimations based only on time of day are not sufficient. There was also no clear distinction in devices discovered on weekdays and weekends. This was further elaborated in [WSM07], where the traced data was used as a basis for an adaptive discovery mechanism (See figure 2.3).

[NM07] also considered commonality, which is computed between subsequent pairs of time. Considering two times t_i and t_{i+1} and A_i as the set of users discovered at time t_i, the commonality is given as the fraction of devices seen at both timeslots:



(a) Contact Time

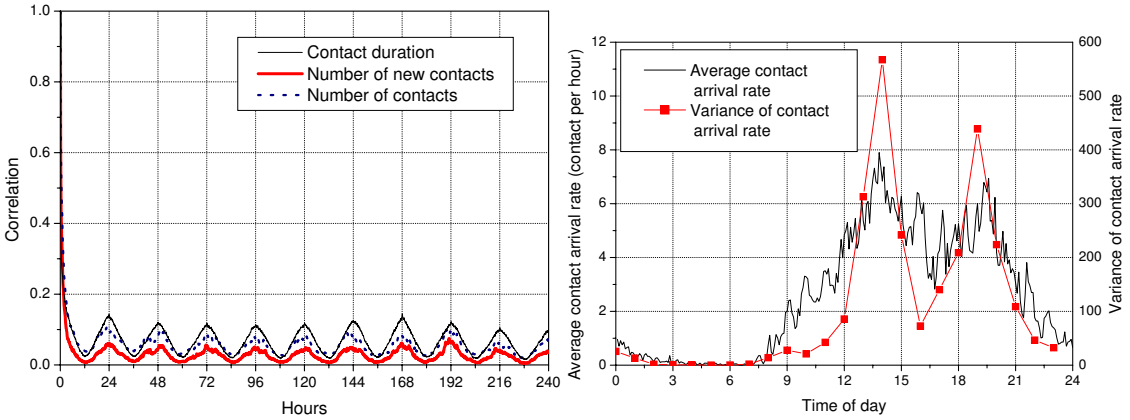
(b) Inter-Pair-Contact Time



(c) Inter-Contact Time

(d) Meeting Time

Figure 2.2: Plots of metric distribution[NM07]



(a) Correlation of contact duration and contact number

(b) Contact arrival rate and variance

Figure 2.3: Correlation of contact duration and contact number over long periods of time (left) and contact arrival rate and its variance over time of day (right) [WSM07]

$$C_i = \frac{|A_i \bigcap A_{i+1}|}{|A_i \bigcup A_{i+1}|} \tag{2.1}$$

The timescales considered are (1) subsequent hours of the same day (HSD), (2) subsequent days (SD), (3) specific times on subsequent days (STSD), (4) specific days of the week as well as (5) specific times on the same day of the week (TDW). The fraction of identical devices seen on these timescales are shown in 2.1, which indicates that the chances of discovering the exact same device again decreases when more time has elapsed since the last discovery of this device was made.

Time Scale	Maximum Commonality
HSD	0.2
SD	0.14
STSD	0.07
DW	0.04
TDW	0.03

Table 2.1: Maximum Commonality across different time scales[NM07]

In [CSTC12] the authors investigated regularity in collected data traces of 10 users over two months. They found that participants spent 85$\pm$3% of time staying in one place and only 13$\pm$3% of time moving. Places were identified using a system called SmartDC,

with which locations were identified either by location data from GPS, the network location based on the currently connected cell towers or with WiFi fingerprints. This indicates a certain regularity in human mobility, which some researchers try to emulate with the help of mobility models.

2.2.2 Modeling Mobility

Algorithms in mobile wireless networks can be evaluated by simulating mobility. These simulations use underlying mobility models, which should be as realistic as possible. Mobility models can generally be categorized to be either synthetic or based on real world traces. Synthetic models are usually described by mathematical models, which simplifies theoretical analysis. Trace models are generated from deployed systems that log data such as connectivity or location.

Real world traces like the ones mentioned in 2.2.1 on page 10 are rare and most of them were released only recently. Although synthetic models are used in most instances, they are problematic because they are prone to be unrealistic. This problem is especially apparent when comparing them to real world traces. Initiatives like CRAWDAD[KH05] try to provide traces for researchers to use, to reduce the problem of incorrect synthetic models. This led to the creation of synthetic models based on traces, to simulate real world scenarios[Mus09].

The most prominent purely synthetic models are:

Random Walk Mobility Model: This is the most prevalent model and also the simplest. It models purely random movement. This may also be called *Brownian Motion*.

Random Waypoint Mobility Model: This model extends the random walk model with changes in speed and direction, as well as pauses in movement.

Other similar models are characterised by the nodes being independent of each other and that movement is randomized. An extension of these models includes group mobility. The groups of nodes are linked to each other by a set of equations. These models are also unrealistic, because the underlying group movement is still based on a random distribution. Groups are also hardwired in most cases, without the dynamics of nodes joining other groups in the simulation.

Trace based models try to provide the same statistical properties as real world traces. Researchers of the ETH Zürich devised a model by dividing the campus into squares

and deriving the movement between these areas from WiFi traces[Tud05]. The probability of user movement is represented as a Markov model. This is similar to Weighted Waypoint Model, where certain waypoints have a higher probability of being chosen as the next destination. Another approach studied the influence of user's movements based on the concept of a popularity gradient between Wireless Access Points[JL05, LKJ06]. The evaluation was done with traces from Dartmouth College. One model discussed in [MSKY05] uses empirical data rather than wireless measurements to analyze movement characteristics of pedestrians.

Both studies mentioned in 2.2.1 on page 10 found that most synthesized mobility models predicted an exponential decay in mobility metrics. The power law distribution stands in contrast with this assumption, implying that some of the models may be unable to model real mobility. Karagiannis, et al [KB10] derived analytical results from 6 sets of traces. They verified the power law of inter-contacts time, though they found that beyond about 12 hours, the cumulative distribution function exhibits exponential decay. With the help of an analytical framework, they show that randomized, synthetic models like the Random Waypoint model should not be abandoned. These models are still able to represent characteristics of empirical traces. They also argue that opportunistic forwarding may still be relevant in this time scale of more than half a day, where the decay becomes exponential, and some forwarding schemes may therefore be too pessimistic.

Another approach to modeling mobility is based on the fact that mobile devices are usually carried by humans, which enables the use of social networking concepts. Models based on social network research are often based on traces. Social networks are characterised by a high level of clustering, which is not observed in other types of networks[New03].

A mobility model based on social network theory is described in [MM06], where the social network is a key input in creating a synthetic network structure. Movement and mapping into a topographical space is driven by social relationships between individuals. Definitions of different types of relationship during a certain time may also be incorporated, to model the fact that work relationships may be more important during work hours, for example.

In [EKKO08], a "Working Day" movement model is presented, which is able to closely reproduce inter-contact times and contact times found in traces. The model is verified and implemented using the ONE simulator[KOK09]. This model contains several submodels, from which one is chosen according to parameters. Submodules mentioned

in the paper are, for instance, staying at home, working and evening activities. The basic premise is that nodes modelled this way will chose a suitable submodel in certain situations. For instance, the activity model for a nodes home will wake it up in the morning, initiate a transport model for the node to go to work, which will be followed by a decision if the node will move back home or do an evening activity.

2.2.3 Predicting Mobility

Mobility prediction is related to the way human mobility is modelled, but rather than reproduce the significant metrics, a mobility prediction algorithm will try to infer future locations based on these mobility metrics.

Anticipating user mobility is useful in a number of applications. It may be used in context-aware applications to infer relevant information depending on the users location. Some opportunistic routing algorithms also use knowledge of future locations to determine the best forwarding strategy. It is also possible to proactively adapt duty cycling of sensors to changes in location, and to predict contact opportunities in a spatial or spatio-temporal manner.

Several methods of prediction have been proposed, with 4 of those being evaluated in [SKJH06]. Here, the predictors are compared using a dataset from more than 6000 users, collected over 2 years. The dataset records registration of users to WiFi Access Points without any additional knowledge of the users mobility. All predictors are compared on their accuracy to predict the users movement to the next cell, where each cell is defined as the AP the user is registered to. The paper referes to each of these cells as a location, although there is no knowledge of the actual location of a user (e.g. longitude and latitude). For each user, a location history is constructed as a series of location changes, or *moves* between different APs. These movements are domain-independent in a way that they do not include the APs location or even timing information of a location change.

The paper discusses Order-k Markov predictors, LZ[1]-based predictors and two predictors called PPM[2] and SPM[3]. The latter three are based on text compression algorithms, due to the fact that they try to predict the next occuring letter based on previously encountered patterns. For the Markov predictors, values of 1 to 4 for the order k have

[1]These are based on an incremental parsing algorithm by Ziv and Lempel
[2]Prediction by Partial Match
[3]Sampled Pattern Matching

been studied, meaning that the prediction was based on a series of up to 4 previous locations. Additionally, a fallback mechanism for the Markov predictors *O*(2) and above was used. In the case of no prediction, an *O*(*k - 1*) predictor would be used recursively until a prediction is made or *O*(1) predictor failed to give a prediction. A predictors accuracy is measured through the possible outcomes of a prediction:

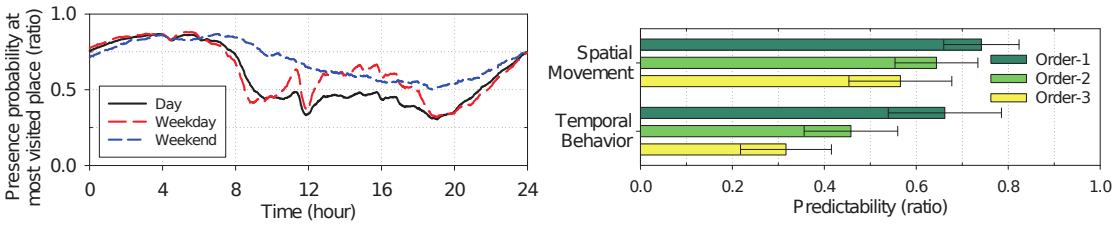
- The predictor correctly identified the next location,
- the predictor incorrectly identified the next location, or
- the predictor failed to return any location.

The accuracy is the fraction of correct predictions, which means that an incorrect prediction as well as no prediction are considered to be incorrect.

One basic characteristic of these predictors is that they faired poorly on short traces (up to 100 movements), indicating a fairly large learning time requirement. Generally, the lower order Markov predictors worked at least as well as the compression-based predictors, while being less complex algorithmically. The fallback mechanism of higher order Markov predictors also improved their accuracy significantly. *O*(2) Markov was deemed the best predictor overall, due to its simple implementation, relatively small space complexity and best overall accuracy.

More recently, Chon, et. al. [CSTC12] evaluated mobility prediction algorithms on fine-grained, continuous mobility data. Here, the authors explore spatio-temporal behavior of 10 users over a timespan of two months. They used a trace of mobility data with room-level accuracy. Regularity in behavior was found to be high, with people spending 83$\pm$3% of time staying in place and 13$\pm$3% of time moving between places. People also tend to stay in a number of frequently visited places. Stay duration in the top two most frequented places is as high as 83$\pm$12%. Movement toward known places with more than 3 previous visits is 69$\pm$7%. Figure 2.4 on the following page shows the regularity and predictability of human movement observed for the previously mentioned trace of 10 users. A persons movement is considered predictable if a mobility pattern has been observed previously.

Rather than predicting the next location the user will visit, predictors were evaluated on their ability to anticipate stay duration. The study assessed a number of different predictive algorithms, divided into two classes: (1) location-dependent algorithms and (2) location-independent algorithms. For the former, a Markov-based predictor as described before has been used. NextPlace and Jyotish (see section 2.3.3 on page 23) were identified as location-independent models, from which NextPlace was chosen for anal-



(a) Temporal regularity in mobility. For example, during 87% of weekdays, the user spends time at one specific place at 4 AM.

(b) Spatial and temporal predictability using high-order Markov predictors

Figure 2.4: Regularity and predictability in human mobility[CSTC12]

ysis. Generally, the location-independent models only use temporal information while the location-dependent models use only spatial information for prediction.

It is possible to add features to the predictors in order to improve results. These features are useful to lessen the impact of non-predictable cases. For instance, higher-order Markov-predictors could not be able to predict a next location if the sequence of previous locations has not yet been observed. Feature-aided schemes also extract patterns based on additional information, thereby extracting this information from the dataset. In [CSTC12], three types of feature-aided schemes are proposed:

- Time-aided schemes, which consider the time of arrival at a place to predict stay duration. This is only applicable for location-dependent schemes because location-independent schemes already consider time of arrival for their prediction.
- Return probability-aided schemes assume that people exhibit temporal regularity in returning to certain places. For instance, a person might return home in the evening at a specific time, regardless of their current location.
- Fallback schemes will use a different predictor in the case of none-prediction. An O(2) Markov-predictor would use the O(1) predictor as fallback, for example.

The paper concludes that the location-dependent models are better suited at predicting temporal regularity. Furthermore, the context of previous location visits does not aid in predicting the stay duration in the current place. This is notable because as mentioned before, the O(2) Markov-predictor was deemed the most accurate for prediction of future locations in [SKJH06], while the feature-aided O(1) Markov predictors were deemed the best in [CSTC12] for prediction of stay duration.

2.3 Adaptive Discovery

Discovery schemes are an integral part of networks using ad-hoc communication. In order to initiate data transfer to another network node, the node must be known to accept connections and it must be in communication range. Discovery schemes, sometimes also called *hello* or *beaconing* protocols, are widely used in sensor networks, where the sensor nodes often sleep for extended times in order to preserve energy. Some network protocols widely used in computers or mobile devices provide direct support to fnd nearby devices. Often, these device scans or inquiries require a large amount of power and are therefore mostly initiated manually and only when specifically needed. For instance, to transfer multimedia data via Bluetooth, the two participating devices will usually have to be set to inquiry or inquiry scan mode manually. The data will the be transferred, after the partner device has been discovered. Afterwards, Bluetooth will often be deactivated. This direct and active transfer of data does not usually happen in DTNs oder PSNs. Here, the data should be forwarded without user interaction. Of course, this necessitates that the used networking technologies are active most of the time. Device discovery has a significant impact in this situation, which is the reason for developing adaptive discovery algorithms that try to run the costly device discovery only when there is a chance that it will actually discover another device. The following sections will describe some algorithms that can be used to adapt the discovery to certain conditions, or infer the presence of contacts.

2.3.1 STAR

Wang, et al[WSM07] devised a heuristic algorithm called STAR[4], which is designed based on some of the conclusions of the study described in [NM07] (see chapter 2.2 on page 10). They also built a theoretical framework from which different algorithms can be compared. This framework provides a theoretically best discovery interval for a given rate of missed contacts. A contact is deemed to be missed when two devices are in communication range but do not probe for the other device and therefore do not know that it is in the vicinity. For a given rate of possible missed contacts, an optimal probing delay can be found. Algorithms that have a shorter average probing delay for this rate of missed contacts are deemed less efficient, especially in terms of used battery power. The algorithms are therefore optimized based on the average probing delay.

[4]Short Term Arrival Rate

The calculated probing delay is constant, and the discovery process will therefore be started after a fixed amount of time.

The STAR algorithm computes an estimated rate of new contacts to be seen in a short timespan. STAR exploits the observed behavior that new contacts are more often seen fairly close together in time. Conversely, there are times when no contacts are seen for several hours, for example during nighttime. The algorithm may have a really low probing delay during certain times of day, though its average is still low because of long delays during other times. The arrival estimate rises according to a power law when new contacts are seen and falls linearly when none are seen. The algorithm also incorporates the fact that nearly no contacts are seen between midnight and 8AM, and therefore increases the estimated arrival rate at 8AM. This increase counters the fact that the discovery interval will probably increase to the maximum value of 1800 seconds during the night, which would make a discovery in the morning unlikely.

2.3.2 eDiscovery

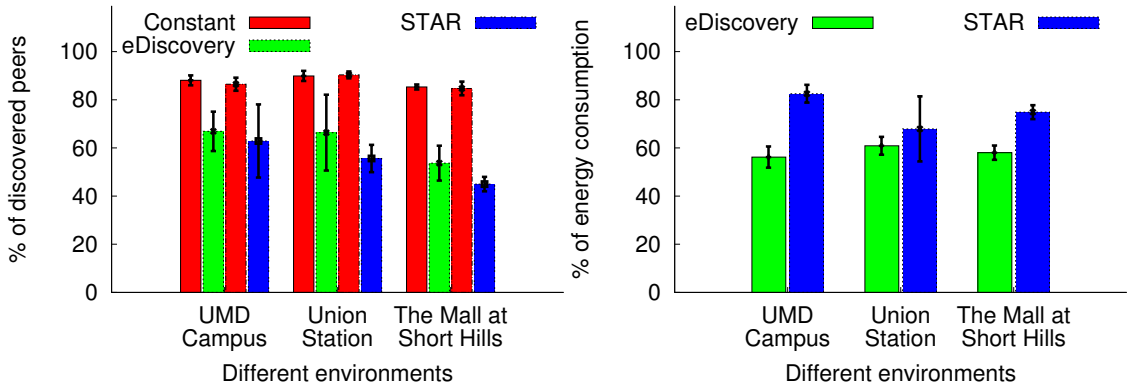
In [HS12] the eDiscovery algorithm is proposed, which aims to improve the energy cost of Bluetooth-based node discovery. The paper describes three approaches for controlling the Bluetooth inquiry process,

- the length of the inquiry window, i.e. the time one inquiry process runs,
- the number of inquiry responses received during the inquiry window, and
- the interval between subsequent inquiries.

The basic Bluetooth discovery process will run for 1,28 seconds, repeated for a specified number of times, which is usually 8 in standard implementations (see section 2.4).

The algorithm focusses on the inquiry window and inquiry interval as parameters, with initial values of 8 for the inquiry window ($8 * 1,28s = 10,24s$) and 10 seconds for the inquiry interval. The inquiry window parameter will remain at 8 as long as the inquiry discovered more than a specified number of peers, designated as N. In case of a lower number of discovered peers, it will be set to $5 + r$, where r is chosen before each inquiry process as

$$r = \begin{cases} 1 & \textit{with probability } 0.1 \\ 0 & \textit{with probability } 0.8 \\ -1 & \textit{with probability } 0.1 \end{cases} \tag{2.2}$$



(a) Ratio of discovered peers

(b) Percentage of energy consumption, in relation to a constant discovery procedure

Figure 2.5: Comparison of eDiscovery and STAR in different environments[HS12]

with the intention of being able to discover the majority of devices within a shorter inquiry time while lowering energy cost.

The inquiry interval is adapted similarly, with the value being increased by $10 + r$ when two consecutive inquiry processes did not discovery any device, and resetting it to $10 + r$ when discovering a peer. The interval is also modified during consecutive rounds of inquiry procedures which discovered peers. When the number of devices discovered in the current round is larger than in the previous one, the interval will be decreased by a predetermined value I. In the opposite case, it will be increased accordingly. The global parameters N and I influence the number of discovered peers and the energy consumption. Lower values lead to lower average inquiry window lengths and inquiry intervals. Basically, these parameters determine the aggressiveness of the algorithm.

In the paper, the algorithm is compared to STAR in three different environments. Each experiment ran for 30 minutes and limited the inquiry interval of STAR and eDiscovery to be between 10 and 200 seconds. In this setup, eDiscovery discovered more peers than STAR while consuming less energy. Figure 2.5 depicts a comparison of the two algorithms, in relation to a discovery scheme that constantly runs the inquiry procedure. In these experiments, the eDiscovery algorithm adapted to the number of discovered peers quicker than STAR.

2.3.3 Jyotish

In [VDN11], the authors present a framework for predicting the movement of people. They try to provide a system to answer three questions they deem fundemental to several research domains, including wireless networks. These questions are: "(1) where will the person stay at a future time (i.e., location)?, (2) How long will she stay at the location (i.e., stay duration)?, and (3) Who will she meet (i.e., contact)?" Their solution is based on a combined WiFi/Bluetooth trace, which is further described in [VNRG10]. The trace incorporates captured MAC-adresses from Bluetooth- and WiFi-devices (mostly Access Points) on the University of Illinois campus. The framework first comprises the WiFi records into locations. The Bluetooth-records are then assigned to these locations using a Naive Bayesian classifier. These records with assigned locations are used as input for a location predictor (question 1), a stay duration predictor (question 2) and a contact predictor (question 3).

The construction of all three predictors is done using two parameters: type of day (v) and time slot (τ). The types of day are $v \in \{weekday, weekend\}$, and each day is divided into timeslots of 1, 2, 4, etc. hours. Given the relation C as the Bluetooth-records with location, each record $r \in C$ is mapped into the type of day and the timeslot, based on this records scan time. Each movement predictor for a person p is a record X where $X = \{v_1, \tau_1\}$. The output is the location, duration at the location and contacts for the type of day and during this time slot. The location and contact predictors use the following Naive Bayesian classifiers:

$$L_X = \underset{k}{arg\,max} \{P(v = v_1 \mid L_k)\ P(\tau = \tau_1 \mid L_k)\ P(L_k)\} \tag{2.3}$$

$$U_X = \underset{j}{arg\,max} \{P(v = v_1 \mid u_j)\ P(\tau = \tau_1 \mid u_j)\ P(u_j)\} \tag{2.4}$$

where L_x and U_x is the most likely location and contact for the input query X, respectively. Both are easily modified to return a number of most likely locations or contacts. The duration predictor is based on the location predictor, assuming it returns the top-k locations. It will predict the stay duration for each of these locations.

The predictors are evaluated based on the dataset mentioned before. Each of them are trained with a set of 80% of the records and then tested with a set of 200 randomly selected records. When using the location predictor to get the 3 most likely locations,

the predictor had an accuracy of more than 80% for about 85% of the nodes tested. According to the researchers, the duration predictor performed considerably well. 80% of nodes had about 60% correct predictions, and 40% of nodes had 80% correct predictions. Like the location predictor, the contact predictor performed better with a larger number of likely contacts. When using the 7 best contact predictions, 80% of nodes obtained more than 70% accuracy, and 60% correctly predicted 80% of contacts.

2.3.4 Other

Drula, et al[DARD07] proposed two different algorithms based on discrete discovery states. Their solution depends on low-level access to the Bluetooth discovery mechanism. They derived 5 states of discovery, based on the scan window, scan interval and residence time in inquiry and inquiry scan mode. These states are computed to optimize either power consumption or mean discovery time, from most aggressive (in terms of energy consumption) to least aggressive. Access to low-level functionality also has the potential to speed up the general discovery process. The two algorithms are based on *recent activity* and *location of past activity*. The first scheme lets the device switch to the most aggressive state when another device is discovered and goes back to less aggressive schemes when no devices are seen for a certain amount of time. The latter scheme depends on a global positioning system to save locations of past contacts. More aggressive modes are used when the device approaches a place where contacts have been seen previously. Both schemes are evaluated in a simulation. For the location scheme, the simulation space is divided into a grid. Each node counts the devices discovered in each of the cells. The discovery state in a cell is chosen as function of the device count in the current cell and the maximum device count of any cell.

Iyer, et al[IL12] proposed an algorithm called Nest which tries to concurrently adjust parameters of the discovery on all participating nodes. Nest contains two components,

1. a consensus algorithm for the beaconing probability, and
2. an estimator for the local neighborhood size.

Each node in the network has an adaptive probability for which it will send a packet at a given time. Each of these packages will contain a node identifier and the aforementioned probability. The consensus algorithm tries to converge the transmission probability to a local optimum. This happens for all nodes that are close to each other. The neighborhood size estimator will then adjust the local values “towards the reciprocal

of the local density of nodes, [...] based on observed statistics of the communication channel"[IL12].

Ingelrest, et al[IMSR07] devised an algorithm that adapts the discovery frequency without needing any additional hardware like GPS. It is based on the assumption that the beaconing should be adapted to the speed of nodes. Their TAP[5]-algorithm adjusts the frequency based on the turnover of nodes. Every node computes the turnover after each discovery process by managing a neighborhood table. The turnover is equal to the ratio of new neighbors to the current total number of neighbors. If the turnover is high, the discovery frequency is deemed too high, because there are not enough changes in the neighborhood table. If the turnover is very low, the discovery frequency should be higher.

A protocol that has been developed to be more efficient than the previously mentioned TAP is proposed in [LM11]. The protocol uses an Autoregressive model to predict the current nodes location, along with the locations of other nodes in the vicinity. Each node divides the time domain into slots, whose length λ is equal for every participating node. At the start of each time slot k, a node samples its own position. Based on the time series of position samples, each node applies a simplified autoregressive (AR) model to estimate its position $\hat{X}_{k+1}$ in the timeslot $k+1$. If the estimate $\hat{X}_{k+1}$ is close enough to the real position X_{k+1}, meaning the estimation error is under a certain threshold, it will be fed as a sample into the AR model. If the estimation error is above the threshold, the samples position will be used instead and node n will send a beacon message, containing its ID and the sampled position. Each node m receiving this beacon will initialize an AR model for node n, estimating its next position. The node n and every receiving node m will therefore have the same estimate and will be updated with new samples positions if the prediction of node n is incorrect. According to simulations, this protocol has a 50% reduced beaconing frequency with a neighborhood discovery performance as high as the one for TAP.

2.4 Node Discovery in Bluetooth

The Bluetooth discovery process is called *Inquiry*. Before establishing a connection, Bluetooth devices usually have to discover each other. The inquiry process depends on two device states, the *inquiry* and the *inquiry scan*. The device that inquires is usually

[5]Turnover based Adaptive HELLO Protocol

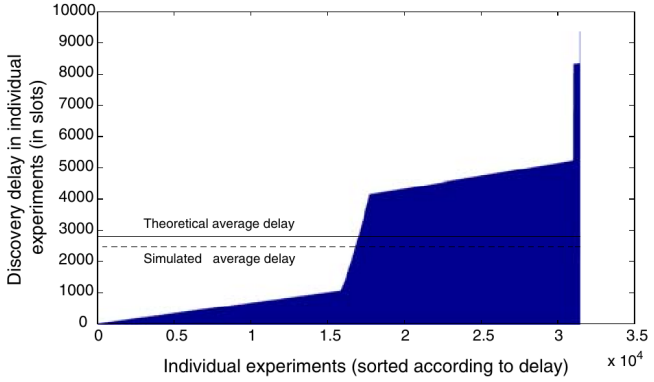


Figure 2.6: Distribution of discovery time[CNC+08]

the one that will initiate a connection to the discovered device, it will therefore be called *master*, while the other device is the *slave*. The master will transmit *identification* (ID) packets and the slave device will respond with *frequency hopping sequence* (FHS) packets when receiving an identification packet.

Bluetooth uses a band of 79 frequencies between 2.402 GHz and 2.480 GHz. 32 distinct frequencies are chosen for the inquiry process, which are then split into two trains of 16 frequencies each. It takes 16 timeslots of 625µs for the transmission of one train. Each train is repeated 256 times before switching to the other train. To minimize error in collecting FHS packets, the trains are switched 3 times, for a total number of 4 trains. The whole inquiry takes $256 * 4 * 10ms = 10.24s$.

A device in the inquiry scan mode will listen on one of the same 32 frequencies mentioned before, but for 2048 time slots. After 2048 slots, another frequency is chosen. If the device receives an ID packet, it will go into its standby or connection state for a random time between 0 and 1023 slots, to avoid collisions of FHS packets. If the device receives a second ID packet after a static backoff period, it will wait one slot, then go into its inquiry response mode, wait another slot and send its FHS packet. Afterwards, it will go back to its inquiry scan state, in order to discover or rediscover devices.

A theoretical and simulated analysis of the discovery protocol in Bluetooth 1.1 has been done by Chakraborty, et al[CNC+08]. The result is, that the mean discovery time is about 2.500 slots, but it varies heavily with many simulations in the range of a few to about 1000 slots, and many in the range of 4000 slots and more, as seen in figure 2.6. Bluetooth 1.2 removed the backoff period of 0 to 1023 slots, which potentially doubled the discovery times in order to avoid collisions, which are unlikely[ZL02].

3 Location-based Discovery Optimization

The previous chapter described the research around DTNs, PSNs, discovery and mobility prediction. This is the basis for the development of a discovery scheme, which exploits location information. This chapter will explain the goals behind the proposed scheme and describe the algorithms used as the basis for it.

3.1 Goals

Maximizing transfer opportunities should be the main goal of discovery schemes. This can easily be achieved with a scheme that constantly probes the environment for nodes to communicate with. The downside to this is the vast ressource needs of discovery processes. Discovery schemes must therefore optimize certain characteristics within ressource boundaries. For mobile applications, especially using modern smartphones, battery power is one of the most valuable ressources. Users expect their mobile phone to run for a certain amount of time before having to recharge it.

The developed discovery scheme for this thesis should therefore be optimized to maximize transfer opportunities with low energy consumption. Additionally, it should discover devices as soon as possible when in communication range, in order to maximize throughput in the case of short contact times. This is especially important in highly mobile environments.

To achieve these goals, the proposed scheme will try to exploit spatio-temporal regularity in a users movement and contacts. Adapting to this regularity should incorporate the previously mentioned metrics. The spatial adaptation will consider previously observed movements of the mobile user, including probable future locations. It will also be optimized to work within energy constraints, by aiming for a low average number of discovery processes over a certain amount of time.

3.2 Relevant Metrics

The following section will analyse the metrics and mobility patterns described in the previous chapter. This analysis will identify the relevant contact and mobility patterns in human mobility scenarios, in order to assess their use in the proposed discovery scheme.

The mobility characteristics described in 2.2.1 exhibit different amounts of usefulness for discovery schemes.

Contact time is a key metric indicating the time during which an inquiry process must be performed in order to discover a device. As mentioned previously, contact times tend to be short. This is especially apparent in high-mobility locations like sidewalks or shopping centres. In these cases, contact times may be as short as a few seconds. In order for a static scheme to discover every device, the time between discovery processes would have to be shorter than the shortest contact time. Schemes that adapt their discovery frequency to be higher when a new device is discovered may miss these contacts easily when the current frequency is too low. This may be the case when a person is on his way to work: The discovery frequency adapted to high inter-contact times, because the person is at home during the night, where new contacts are seen rarely. The mobile device may not discover any devices on the persons way to work, because contacts tend to be short. The STAR-algorithm (see section 2.3.1 on page 20) counters this by increasing the frequency at a predetermined time, like 8AM.

Inter-contact time is useful in predicting future contact possibilities. The power-law distribution of inter-contact times indicates that a majority of contacts are discovered within a certain timeframe. This metric is directly linked to the way the STAR-algorithm adapts its discovery frequency. Whenever a new device is discovered using the current discovery frequency, the frequency will be adapted according to the power law of inter-contact times. The variation in the number of contacts seen during different times of the day leads to differing inter-contact times. Most contacts will be seen in places and at times where many other people are present. For instance, inter-contact times will be high during the night, when a person is alone at home. While algorithms like STAR only consider this distribution in a temporal manner, the proposed scheme aims to adapt to inter-contact times in a spatial context.

The *meeting* metrics are useful to adapt to situations where several devices are discovered in a short amount of time. Similar to the TAP-algorithm described in section 2.3.4 on page 25, the discovery frequency should adapt to the number of previously un-

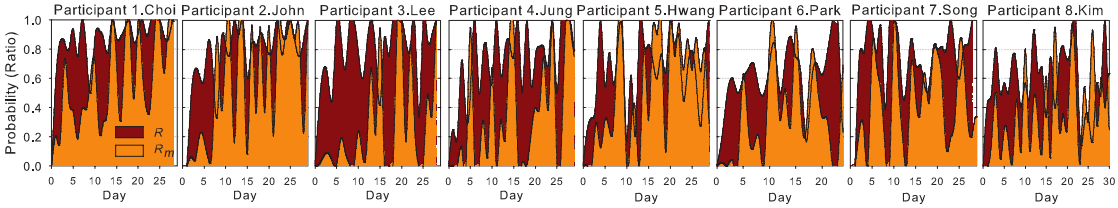


Figure 3.1: Predictability of the participating users in [CTSC11]. Location predictability (R) is higher than residence time predictability (R_m)

seen nodes. A high *average instantaneous meeting size*, for instance, indicates that a new discovery process will probably discovery the same devices as before. Accordingly, a lower number indicates a dynamic environment, where each discovery process finds devices it has not seen before. In a meeting with a lot of devices, the discovery frequency can be lowered significantly if the meeting is static. The average instantaneous meeting size also correlates to contact times in meetings.

Inter-pair-contact time is important in situations where a contact to a certain device is desired. This is the case for some routing schemes which predict delivery probability for each device. A discovery scheme may aim to discover devices with a high delivery probability. The focus of this thesis will be to discover devices in general, rendering this particular metric less important.

Considering location and mobility metrics, the proposed discovery scheme aims to exploit regularity in movement. The most important regularity to consider for this is in visiting specific future locations. The ability to predict movements lets the discovery scheme adapt to specific spatio-temporal conditions. Figure 3.1 represents the predictability observed by Chon, et. al. in an experiment outlined in [CTSC11]. The participating graduate students used the SmartDC system installed on their smartphones to capture location visits and stay duration. In the graphs, $R = \frac{No.of\ revisits}{No.of\ visits}$ represents the revisit ratio while $R_m = \frac{Sum.of\ previously\ observed\ residence\ time\ at\ each\ revisit}{Sum.of\ residence\ time\ at\ each\ visit}$ represents the residence time ratio. As indicated, the revisit ratio is, which leads to a better predictability, while the stay duration is less predictable. Due to the relatively high predictability of location visits, the proposed discovery scheme will include mobility prediction.

Another interesting metric is the commonality described on page 12. The low commonality for the time slots shows that only a fraction of devices are discovered again, for example on the same weekday in 2 consecutive weeks. This could indicate that the number of previously undiscovered unique devices is fairly large. Assuming a person commonly discovers about the same devices when at work (for example those of col-

leagues) and at home, the way from and to work, or to other less frequently visited locations, may see a large influx of unique devices.

3.3 Mobility Prediction

Contact patterns are, like human mobility, predictable to a certain degree, and contact predictability is based on spatio-temporal predictability. These assumptions are based on the fact that contact patterns in PSNs are human-centered. The regularity in people movement, their spatio-temporal preferences and social tendencies may therefore be exploited to improve discovery schemes. The basic strategy of the proposed discovery scheme is linking contacts to the geographical locations where they are discovered. The mobile device is then able to adapt to previously observed contact patterns while moving towards these locations and while staying at them. To facilitate this, the users movements and stay duration are predicted. The predictions are used to adapt the discovery frequency to the users current location.

Mobility prediction depends on the the users regular mobility. It is assumed that there exist certain places, for instance the workplace and home, where the user spends a significant amount of time. There are also several less important places, where visits happen only occasionally or irregularly. Movement between places generally amounts to a relatively small fraction of the users time. On the other hand, these movements may still lead to a significant number of contacts. A person that uses public transportation on the way to work will probably encounter a large number of other people. To summarize, the assumption is that there are places and movements between places where a significant amount of contacts are registered.

To exploit this regularity and adapt the discovery scheme to local conditions, the first step has to be the identification of significant places. This section will describe an approach based on clustering movement and location patterns. The clusters hold information on contacts and mobility, including stay time and movement to other clusters. A mobility prediction approach will be discussed, which uses this information to predict future locations. Finally, an adaptive discovery scheme will be proposed, which uses information from the cluster and information of previous contacts on the way between clusters to improve the discovery.

3.3.1 Clustering Locations

Identifying important locations can be done with a clustering algorithm. Clustering is mostly used to categorize a large amount of data, based on some similarity metric. Clustering in this thesis will be used to facilitate the mobility prediction and to identify general spatial regions the user stays in. For these regions, certain mobility and contact metrics can be identified individually, to adapt the discovery scheme to local characteristics.

In [ZZXY10], a clustering algorithm for identifying stay regions is described, which will be used for clustering locations in this thesis. The paper defines the algorithm in terms of trajectories, stay points and stay regions.

Trajectories T are sequences of location points, defined as $T = \langle p_0, p_1, \ldots, p_k \rangle$, with the location points $p_i = (x_i, y_i, t_i), \forall 0 \leq i < k$ containing a timestamp t_i and the two-dimensional coordinates (x_i, y_i).

Stay points s are regions where a user stayed longer than a time threshold T_s and within a distance threshold D_s. Stay points are comprised of a set of consecutive points within the users trajectory, where each point is within the threshold distance of each other and the elapsed time between the first and last point is longer than the time threshold. In other words, denote a set of points $P = \langle p_m, p_{m+1}, \ldots, p_n \rangle$, where $\forall m < i \leq n$, $Dist(p_m, p_i) \leq D_r$, $Dist(p_m, p_{n+1}) > D_s$ and $Int(p_m, p_n) \geq T_s$. $Dist(p_i, p_j)$ is the geospatial distance between two points, $Int(p_i, p_j)$ is the time difference between the two points timestamps. A stay points coordinates are averaged over the coordinates contained in the set:

$$x_s = \sum_{i=m}^{n} \frac{x_{p_i}}{|P|}, y_s = \sum_{i=m}^{n} \frac{y_{p_i}}{|P|} \tag{3.1}$$

The arrival and departure times are represented as the timestamp of the first ($t_s^a = t_{p_m}$) and last point ($t_s^d = t_{p_n}$) in the set, respectively.

Stay regions r are used to cluster several stay points into geographical regions. A clustering algorithm is used to extract a number of stay regions. The algorithm uses all extracted stay points $S = \{s_1, s_2, \ldots, s_N\}$ as input to generate geographic regions containing the stay points $S' = \{s'_m, s'_{m+1}, \ldots, s'_n | s'_i \in S, \forall m \leq i \leq n\}$ belonging to the same cluster. Stay regions also have their coordinates averaged over the

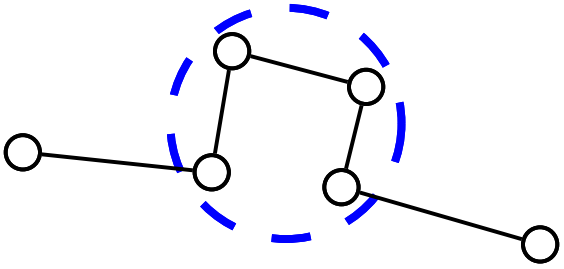


Figure 3.2: Stay point (blue) generation from trajectories (black)

containing stay points coordinates:

$$x_r = \sum_{i=m}^{n} \frac{x_{s_i}}{|S'|}, y_r = \sum_{i=m}^{n} \frac{y_{s_i}}{|S'|} \tag{3.2}$$

The clustering algorithm used in [ZZXY10] is based on dividing the map into a grid. A grid-based layout with predefined grid sizes constrains the size of clusters, which would not be the case for other clustering algorithms like k-means clustering or OPTICS [SJ03]. Each grid is defined as a square of width $\frac{d}{3}$. All detected stay points are projected into the grid, according to their location. Afterwards, grids are clustered by choosing the grid with the most stay points that has not been assigned to a stay region. Each grid adjacent to it that has neither been assigned a stay region is clustered with the chosen grid into one stay region. This process is repeated until there are no more grid containing stay points. This ensures that each stay region has a maximum size of $d \times d$. Finally, the stay regions coordinates are computed as shown in the equations 3.2. This clustering algorithm has the disadvantage that it is not incremental. When the number of stay points grows, some regions may not be the most significant anymore. In this case, stay regions could be recreated, which may require a significant amount of computing power.

Rather than the grid-based approach, the clustering used in this thesis will be based on clustering stay points according to their geospatial distance, similar to the way stay points are computed. To extract a number of significant stay regions, each stay region must contain at least N stay points. This way, only regions that have been visited a minimal number of times are considered for the mobility prediction. Stay regions are defined as a a set of stay points $S = \langle s_1, s_2, \ldots, s_n \rangle$, where $\forall 1 \leq i \leq n$, $Dist\,(s_m, s_i) \leq D_r$ and $n \geq N$. Each stay point can only belong to one cluster, and new clusters can be created incrementally.

The clustering algorithm therefore works as follows: Every new location point is compared to the one before. If they are within the distance threshold D_s, a new possible stay point is created. All following location points are checked for their distance to the possible stay point. As soon as one location point is outside the threshold, the possible stay point is checked for the stay time. If it is longer than T_s, the stay point is saved. Otherwise, it will be discarded. If the stay point is saved, its distance to the existing stay regions will be compared. If it lies within distance of any of them, it will be assigned to the closest stay region. If it is not within a stay region, the distance to every other stay point which is also not currently part of a stay region is computed. If there are more than N stay points within a distance of D_r, a new stay region is generated.

3.3.2 Predicting Mobility

The basis for the mobility prediction described in this thesis is the users location history. The location history is defined as a consecutive list of stay regions $H = \{r_1, \ldots, r_n\}$. From the sets of stay points S and stay regions R, the movement between two stay regions r_1 and r_2 can be extracted as a list of stay points $S' = \{s_1, \ldots, s_n\}$, where $s_1 \in r_1$, $s_n \in r_2$ and $\forall i | 1 < i < n, s_i \notin R$. That is, the first stay point is located in the first stay region, the last stay point is located in the second region, and each stay point in between is not part of any region. Generating these movements from the list of stay points in consecutive order will provide the history of movements. Because the clustering algorithm works incrementally, stay points between two previously registered regions may become part of a new region. This would lead to a necessary reconstruction of the entire movement history. For the sake of simplicity, the algorithm as presented here will not recreate the mobility history each time a new stay region is generated.

After constructing the movement history, a mobility prediction algorithm can be used to infer future location visits. These predictions allow the discovery scheme to adapt its discovery frequency to current and future conditions. Markov-based prediction algorithms are widely used because they provide good accuracy while being easy to implement, as described in section 2.2.3 on page 17. In this thesis, the Order(1) Markov predictor will be used. While the O(2)-predictor might be more accurate in some circumstances, it has a higher failure rate due to the fact that a larger number of recent location sequence may not have been encountered before. Maintaining a low failure rate would necessitate a fallback-scheme to the O(1) Markov predictor, increasing the amount of computation needed to predict a location.

Next location	Probability
a	$3/6$
b	$0/6$
c	$3/6$
d	$0/6$

(a) Probabilities for the $O(1)$-Markov predictor

Next location	Probability
a	$2/6$
b	$0/6$
c	$3/6$
d	$1/6$

(b) Probabilities for the $O(2)$-Markov predictor

Table 3.1: Markov predictor probabilities for the location history H=*{ababcadabcababcbab}*

Markov schemes use the context of the recent k locations $L\left(n-k+1,n\right)=l_{n-k+1},\ldots l_n$, extracted from the location history H, to predict the next location. The order-k Markov-predictor, using the sequence of last k locations including the current location, computes the probability that the next location will be l_{n+1} as follows:

$$P_k\left(l_{n+1}\right)=\frac{N\left(L\left(n-k+1,n+1\right),H\right)}{N\left(L\left(n-k+1,n\right),H\right)} \tag{3.3}$$

Here, $N\left(L,H\right)$ denotes how often the location sequence L has been encountered in the location history H. Consequently, the probability will be $\frac{0}{1}=0$ if the context $L\left(n-k+1,n\right)$ has never been encountered before. In this case, the Markov-predictor is unable to give any prediction. Otherwise, the location with the most probability to be visited next can be found. For example, considering the location history $H=\{ababcadabcababcbab\}$, the $O(1)$ and $O(2)$-Markov predictors would compute the probabilities for the next location shown in table 3.1, based on the current context of $L\left(b\right)$ and $L\left(a,b\right)$, respectively. As shown, the next location would most probably be location c, although the $O(1)$-predictor predicts the same probability for location a.

To ease the computation of probabilities, a graph can be created, where the locations are graph nodes and each edge is a movement from one location to the next. The number of movements can be attributed to the edges. For the location history mentioned before, the mobility graph for the O(2) Markov predictor would look like the one in figure 3.3.

The location prediction will usually happen when a person just left a stay region. In this case, the algorithm will try to predict the next location. Contacts may be mapped to the transition between the region the user just left and the probable next region, as well as the regions themselves. This allows the discovery scheme to adapt to transitions

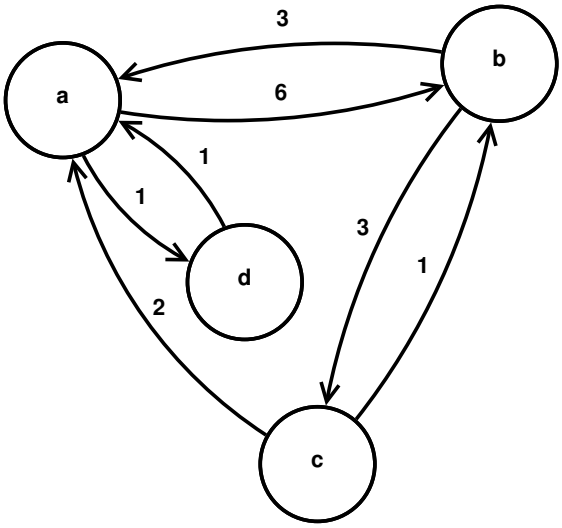


Figure 3.3: Mobility graph with a context of 1 for the location history *H={ababcadabcababcbab}*

and regions. The mapping of location will be described next.

3.3.3 Mapping Contacts to Locations and Paths

In order to be able to adapt to stay regions and movements between them, contacts have to be associated to the place they were registered at. The discovery scheme relies on spatial and temporal information, which is why contacts are defined as $c = \{x, y, t, I\}$, with x and y as the coordinates of the location this contact has been registered, and t as the time the contact occured. I may contain additional information, most importantly an unique identifier, which is used to differentiate the contacts between those to newly discovered devices and those to devices discovered before.

Based on the location and time information, a contact can be directly mapped to a location point. Consequently, the contacts registered in stay points are those mapped to the location points contained in it. A number of contacts will also be registered outside of the stay points. These contacts are easily mapped by considering the time they are registered. For a movement between two stay points s_1 and s_2, the list of contacts is defined as $C_{s_1 s_2} = \{c_1, \ldots, c_n\}$, where $\forall 1 \leq i \leq n, c_i^t > \mathit{Departure}(s_1)$ and $c_i^t < \mathit{Arrival}(s_2)$, with the departure and arrival times of the stay points being the timestamp of their last and first location point, respectively. Contacts in stay regions will be the those made in and between all the stay points contained in the region. Because of the fact that stay

regions and stay points are created incrementally, the number of contacts previously observed does not need to be queried each time the user changes location or arrives at a new stay region.

The list of contacts in stay regions and in movements between the stay regions have to be put into a form usable by the discovery adaptation. The number of contacts in the list, for instance, does not provide enough information on the actual distribution of contacts in a region. The discovery will be adapted to the contact distribution in stay regions and in movements between regions based on the current time. The contact distribution for stay regions and paths will be calculated as the ratio of contacts discovered on an hourly basis. The distribution or each stay region and each transition between stay regions is thus divided into 24 buckets. For every hour h, the contact distribution is $D(h) = \frac{No.\,of\,contacts\,discovered\,in\,h}{No.\,of\,contacts\,discovered}$. This allows the discovery algorithm to adapt to the ratio of devices previously discovered at the current time, in the current location.

3.4 Discovery Adaptation

The adaptive discovery proposed in this thesis is largely based on the eDiscovery algorithm described in section 2.3.2, while incorporating insights from human mobility metrics and the STAR algorithm. The eDiscovery algorithm only varies the inquiry window and the inquiry interval, while all other values are statically chosen. In particular, the minimum and maximum values for the inquiry interval were set to be between 10 and 200 seconds. These values were prudent for the environments outlined in [HS12], especially due to the fact that each experiment only lasted for 30 minutes. STAR, on the other hand, was evaluated over much larger timespans, and its relatively high energy consumption during the day was outweighed by high inquiry intervals of up to 1800 seconds during the night[WSM07]. This is not accounted for in the evaluations made by Han[HS12], indicating that some of STARs assumptions are still relevant in certain situations and over larger timespans.

In eDiscovery, the adaptable parameters are the inquiry window and the inquiry interval. For some systems, most notably the Android operating system, the inquiry window can not be changed by default. In the case of Android, an application would have to get "root" privileges from the system, but most Android phones do not provide these privileges. The adaptive discovery proposed in this thesis will therefore use the default inquiry window of 10.24 seconds and focus on adapting the inquiry interval. To improve on the basic premise of eDiscovery, the proposed discovery scheme will adapt

Algorithm 3.1 Calculate the new inquiry interval based on the previous interval and the current contact distribution

```
1  def calculate_interval(interval, contact_distribution):
2    peers = inquire()
3    if peers is 0 and last_peers is 0:
4      interval += 10 + r
5    elif peers is not 0 and last_peers is 0:
6      interval = 10 + r
7    elif peers > last_peers:
8      interval -= I
9    elif peers < last_peers:
10     interval += I
11
12   max_interval = i_low + contact_distribution * (i_high - i_low)
13
14   if interval > max_interval:
15     interval = max_interval
16
17   last_peers = peers
18   return interval
```

the value chosen for the minimum and maximum inquiry interval, which eDiscovery set to 10 ± 1 and 200 seconds, respectively. This interval will be adapted based on the calculated contact distribution, as described in the previous section. By doing this, the scheme can adapt to situations where contacts are rarely seen in a dynamic way. This is more flexible than STAR, which increases the expected number of incoming contacts at a specific time, namely 8 AM. The proposed scheme, on the other hand, will adapt to lower intervals when the contact distribution changes, which may be at 8AM for people who go to work at this time. Yet, for people who work night shifts, this may not be the case. The discovery scheme also aims to be more dynamic than eDiscovery, which sets the maximum inquiry interval to fairly low values. In situations where no contacts are seen for a longer duration, the interval will still be relatively high, thus consuming more energy than necessary. When increasing the maximum interval of eDiscovery to higher values, it may not be able to adapt to sudden changes in the contact distribution.

The actual algorithm to compute the discovery interval is nearly identical to eDiscovery. The two differences are:

1. The inquiry window will not be adapted by default. This will also make the algorithm adaptable for technologies other than Bluetooth, which may not be specified to have a variable inquiry window. The proposed scheme still allows the adaptation of the inquiry window in situations in which the technology and sys-

tem would be able to.

2. The maximum inquiry interval will be based on the contact distribution at the current location or on the current path. The path to which the scheme adapts to will be the one predicted the mobility prediction algorithm. The location to which the scheme adapts to will be the one the device is currently located in. The locations are equivalent to the stay regions described previously, while the paths are those between the stay regions.

The current maximum interval will be allowed to vary between two values I_{low} and I_{high}. The actual maximum interval will be calculated as $I(h) = I_{low} + D(h) * (I_{high} - I_{low})$, with $D(h)$ being the calculated contact distribution at the current location or path, at the specified hour. Simply put, the maximum interval will be reciprocally proportional to the contact distribution. Algorithm 3.1 shows the basic algorithm. It will adapt the interval similar to eDiscovery, but will set the maximum interval based on the calculated value. The values I and r are the same as those described for eDiscovery in section 2.3.2. I can be set to different values to several different values to make the algorithm more or less aggressive.

4 System Design and Implementation

In this chapter, the logging application used to log mobility and contact data will be described. The description will begin with the basic, event-based data model used in the application. The following section will provide the data model used to store the mobility data, including stay regions and stay points. This data will be put in context by a description of the general application architecture. This architecture is built to be integrated into the Android operating system. Along with the architecture, the implementation of the discovery scheme will be outlined.

4.1 Logged data

The system uses an event-based approach to capturing all necessary data, including contacts and location information. Each event consists of a location with a timestamp, the actual time the event was recorded and event-specific additional information. This is direct a representation of the way the Android system handles certain kinds of technologies. For Bluetooth, the start and end of a discovery process are registered as seperate events. Discovering a device during this time is also its own event. WiFi scans are

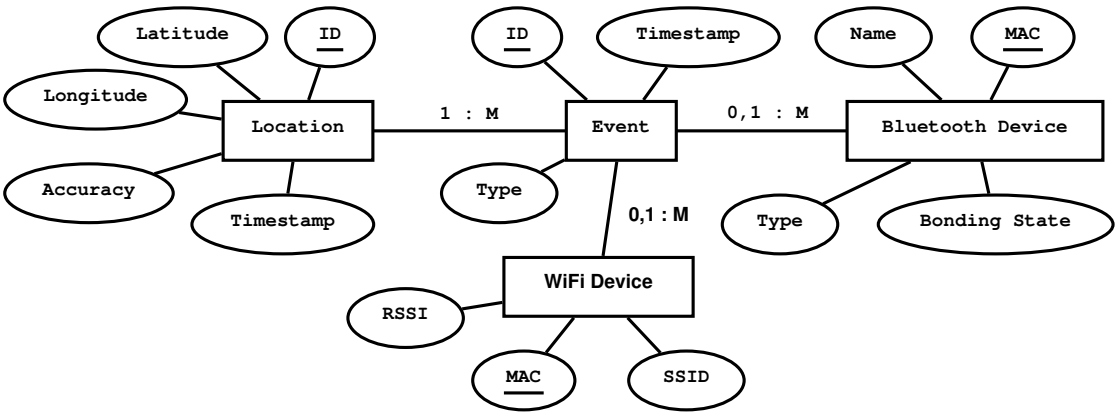


Figure 4.1: Entity relationship diagram of mobility and contact data

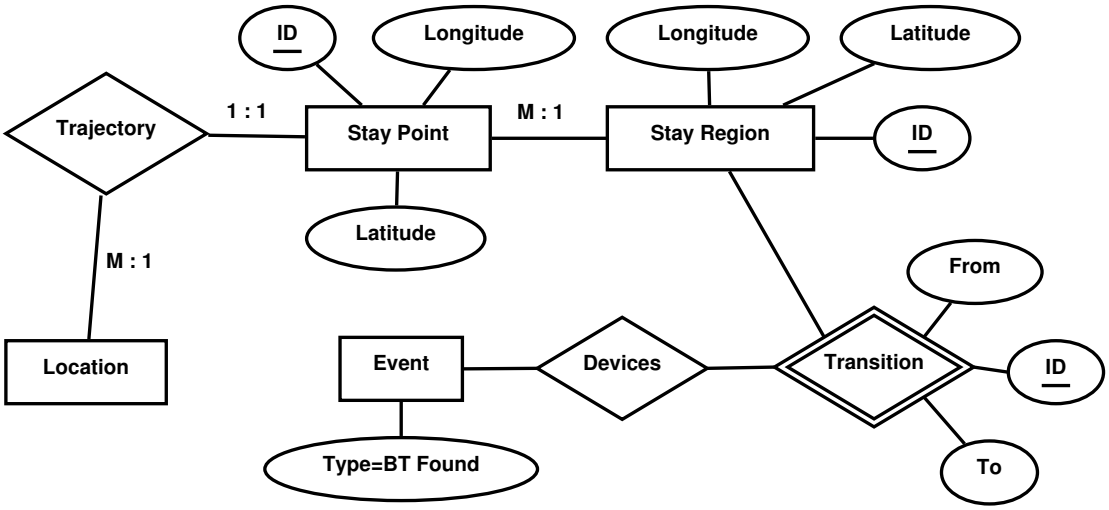


Figure 4.2: Entity relationship diagram for stay points and stay regions. Location and Event attributes omitted for brevity, see figure 4.1 on the previous page

handled differently, where the application is only informed when the scan is finished by receiving a list of nearby devices. Location information is captured by the system in the background. Applications can register for updates from the Mobile Positioning System (MPS), which will send location information as soon as it is available. For this reason, the system captures the time of the actual event and the time of the last location fix seperately.

As seen in figure 4.1 on the previous page, there are no entities for the Bluetooth discovery process. These events do not contain any additional data and are therefore only distinguished by their event type. In order to capture only location information, for use in the mobility clustering and prediction, an additional location event type is specified. The accuracy attribute of locations is provided by the Android system. The accuracy is mainly dependent on the positioning system in use, with network positioning usually providing accuracy of up to 20 metres and GPS reaching accuracies of a few metres. Due to the clustering used in the system, the locations do not need to be overly accurate. The mobility prediction algorithm only considers fairly large regions.

4.2 Clustering and Mobility Prediction

Based on the captured location points, stay points and stay regions are computed. With each new event, the captured location is compared to the last known location. If the

current location is within the specified distance threshold of the last location, a new trajectory is generated. New location points are added to this trajectory as long as the location is within the bounds of the trajectories first location. As soon as the location is outside of this boundary, the time threshold is checked. If the time difference between the first and last locations in the current trajectory is higher than the time threshold, a new stay point is generated. For this stay point, the location is computed once as the average of the trajectories locations. Additional information, like the arrival and departure time at this specific stay point, can be computed from the trajectories first and last locations, respectively. Contact times and locations in stay regions and stay points are associated via the location table. For each trajectory in a stay point, the contact events can be selected that correspond to these locations.

Because contacts could happen in stay regions, but their location point is not part of a stay point, a stay regions contacts will be selected and cached based on their location information. To enable a fast mobility prediction, transitions between stay regions are saved in a seperate table, as soon as they are discovered. Furthermore, to accomodate all contacts made in transition between stay regions, transitions reference bluetooth contact events in a seperate association table.

4.3 Application Architecture

The application uses several features of the Android API to provide extensibility and to allow other applications to benefit from it. The basic components of Android applications are *Activities*, *Services*, *Content Providers* and *Broadcast Receivers*[1]. Activities are generally what a user sees when running an application. Each activity is usually one screen with which the user can interact.. Services provide a way to run operations in the background, even when no activity of the application is currently visible to the user. Content providers grant access to application data for the application itself or for other applications. They may use several different backends for storing the date, including files, SQLite databases or even servers accessed over the network. Lastly, broadcast receivers respond to events from the Android system. These events include state changes or events from wireless technologies, as described in section 4.1 on page 39.

By using the previously mentioned components, the application is able to be integrated

[1]Further information is available at `http://developer.android.com/guide/components/fundamentals.html`

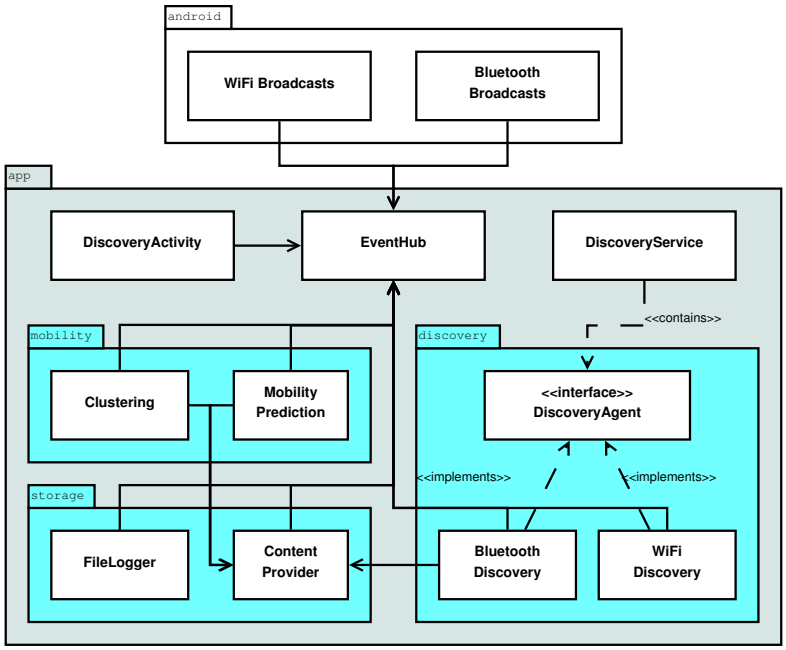


Figure 4.3: Application architecture

into other systems by providing mobility and contact traces, as well as a dynamic discovery approach. The applications architecture is depicted in figure 4.3.

The *Discovery Activity* is the main entrance point of the application. From here, the *Discovery Service* and the *Configuration Activity* can be started. Configuration options include choosing different discovery schemes, which include the one proposed in this thesis as well as static schemes, which start the discovery or scan process in a fixed interval. It is also possible to upload the files created by the *File Logger* to a server. Additionally, the discovery activity shows a list of events that have recently been registered.

The *Discovery Service* is started by the main activity and runs in the background until it is explicitly stopped. It will initialize a list of *Discovery Agents* which have been designated by the configuration activity. *Discovery Agent* is a simple marker interface, requiring no methods to be implemented by classes marked by the interface. All communication between components is based on a system similar to publish-subscribe, which will be described later. The service also initializes the *File Logger* which writes all events to files on the local filesystem.

The *Content Provider* is the main component to provide access to the event data to the application. It publishes an interface to the data described in sections 4.1 on page 39

and 4.2 on page 40 in a SQLite-database. Other components are able to use this interface to store and access the basic mobility and contact data, as well as the computed data including stay points and stay regions. Using a content provider also enables other applications installed on the system to access this data. This is especially useful for applications implementing DTN-functionality. These applications are able to implement routing based on the provided data, without the need to gather this data themselves. The *storage* package also contains a simple class that logs all registered events to local files.

All classes related to clustering and mobility prediction are contained in the *mobility* package. These classes directly access the content provider, in order to store and access registered stay points and stay regions.

Communication between the aforementioned components is done via the *Event Hub*. This hub also provides an interface to the events generated by the Android system. As a matter of fact, the event hub is a broadcast receiver, registered to receive all system events related to WiFi and Bluetooth communication. These events are routed through a *Local Broadcast Manager* to which most of the other application components register. For instance, the *Bluetooth Discovery Agent* will register for bluetooth events like the start and end of a discovery process, as well as the discovery of a new device. Most events used to communicate between components of the application will in turn be routed to the global *Broadcast Manager*. This includes predictions about future locations, generated by the mobility predictor. A prediction event will be used by the bluetooth discovery class to adapt its discovery. Similar to the way the content provider lets other applications access the data, these broadcasts will also inform other applications of the current state of the discovery application. Basically, another DTN-application only has to start the discovery service implemented here and listen to global contact events in order to take advantage of the adaptive discovery.

The discovery scheme is designed to be integrated into the application as a discovery agent. Figure 4.4 depicts the general data flow of the application. The upper part is used for sensing and integrates the positioning system as well as the Bluetooth discovery agent. Every sensing event, from locations to bluetooth discoveries, will be used by the clustering algorithm. The clustering algorithm creates clusters on the fly. Every event will include a location, from which the clustering creates stay points and stay regions. The clustering components will also check for mobility, by comparing each new event location against the current location. These movements are relevant in two cases: Entering a region and leaving the current region. If the user leaves the current region,

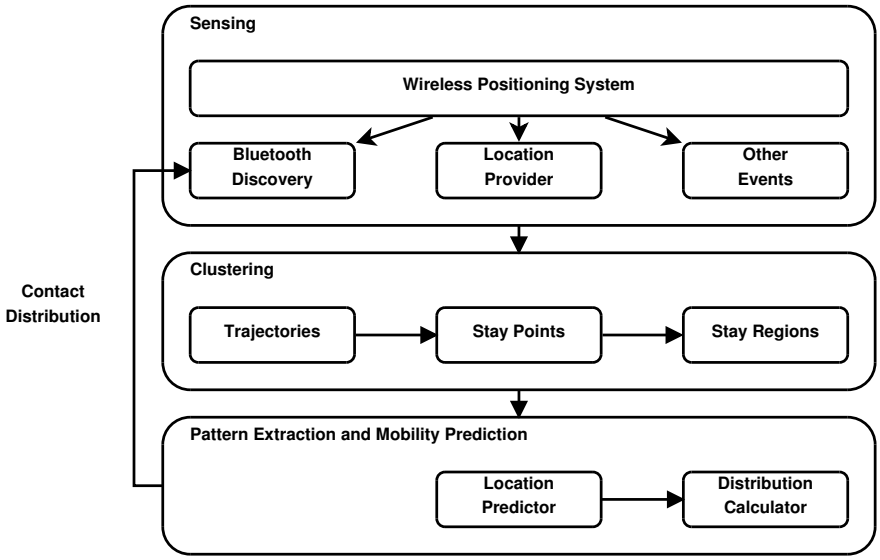


Figure 4.4: Basic operation cycle for the adaptive discovery scheme

the mobility prediction is triggered. The Markov predictor will access the stored transition data from the content provider and calculate the most probable next location. This location is used by the distribution calculator, along with the current hour of the day, to compute the current contact distribution. The contribution, in turn, will be used by the Bluetooth discovery to calculate the maximum inquiry interval for the current location at the current time, effectively closing the circle. The distribution calculator will also be informed when the user enters a stay region. Additionally, the clustering component updates the transitions between stay regions with the transition from the previous location to the one the user just entered. This transition includes all contacts made during the movement, so that the distribution calculator can compute the distribution the next time this specific transition is predicted.

Some other events may also trigger certain updates. If the hour of the day changes, the distribution calculator will update the distribution, which will also update the maximum inquiry time of the Bluetooth discovery.

5 Evaluation

An early version of the Android application was used to capture real mobility data to test the algorithms. The application logged all Bluetooth and WiFi events, along with the location reported by Androids WPS, to local files. The data was gathered using static discovery algorithms, which started the WiFi device scan and the Bluetooth discovery every 30 seconds. Table 5.1 shows the amount of different events registered and different kinds of Bluetooth devices encountered during nearly 3 months of running the application on one device.

Note that the number of WiFi scans is significantly higher than the number of Bluetooth inquiries, despite the fact that the application itself only starts both of them in an interval of 30 seconds. This discrepancy is explained by the way Android handles the WiFi scanning under certain configuration options. In the case of the mobile phone used, Android was configured to scan for open WiFi Access Points, which the Android system seems to perform in an interval much smaller than 30 seconds. The results of these scans were also registered by the application. Another factor is the way the developed application handled the interval between subsequent scans. The scanning and inquiry procedures were scheduled to run in 30 seconds from the time the last scan finished. WiFi scans typically take about one second to finish, while the standard Bluetooth inquiry runs for 10.24 seconds, as described in section 2.4.

The device type was included to get an idea of the fraction of devices that would be capable to participate in a PSN. In general, this would be the case for mobile phones and laptops, although the latter mostly experience a much smaller amount of mobility. As can be seen in the table, 90% of unique devices discovered during the data gathering were in fact mobile phones. Of course, mobile phones are only discovered when they are set to be in inquiry scan mode. This is usually not the case for Android phones, which have to be set to be discoverable manually or through an application explicitly telling the system to be discoverable. The widespread use of smartphones, with Android being one of the most widely used smartphone operating systems, means that this number could potentially be much higher. The standard mode of not being discov-

Event	Number of events
Bluetooth inquiries	91.960
Bluetooth device found	17.631
Unique Bluetooth device found	2.735
WiFi scan	1.017.097
Unique WiFi device found	13.454
Location points	261.656

(a) Number of events registered

Device Type	Number of Devices	Percentage
Phone	2468	90 %
Computer	127	4,6 %
Other (Audio/Video, Toys, etc.)	140	5,6 %

(b) Bluetooth devices encountered

Table 5.1: Data acquired over a period of about 3 months from one device

erable will probably not be changed until DTN and PSN applications become standard on mobile devices. The increasing amount of mobile network traffic, along with the possibility to lessen the strain on the infrastructure with DTN applications, may change this in the future. On the other hand, the number of WiFi Access Points discovered is much larger, and WiFi provides a higher bandwidth. Some mobile phones are also capable of using WiFi in an ad-hoc mode, similar to the way Bluetooth is used currently. Using WiFi in certain circumstances may be more desirable. Bluetooth is still in active development, with the latest specification with version number 4.0 released in 2010[1]. Since version 3.0, Bluetooth also provides data transfer over a different physical layer, including WiFi. This enables Bluetooth to be used for device discovery and session initiation, while a technology with more bandwidth is used to transfer data.

The following sections will evaluate the algorithms described in this thesis by using the captured data. Figure 5.1 shows all locations where a Bluetooth device was discovered. There are easily identifiable regions and paths where contacts seem to be common. In fact, the large number of contacts seen slightly left of the upper middle part of the map is contains the home location of the person who carried the device. The lower right corner is the approximate location of the workplace and university of this person.

[1]See http://www.bluetooth.org/Technical/Specifications/adopted.htm for all Bluetooth specifications

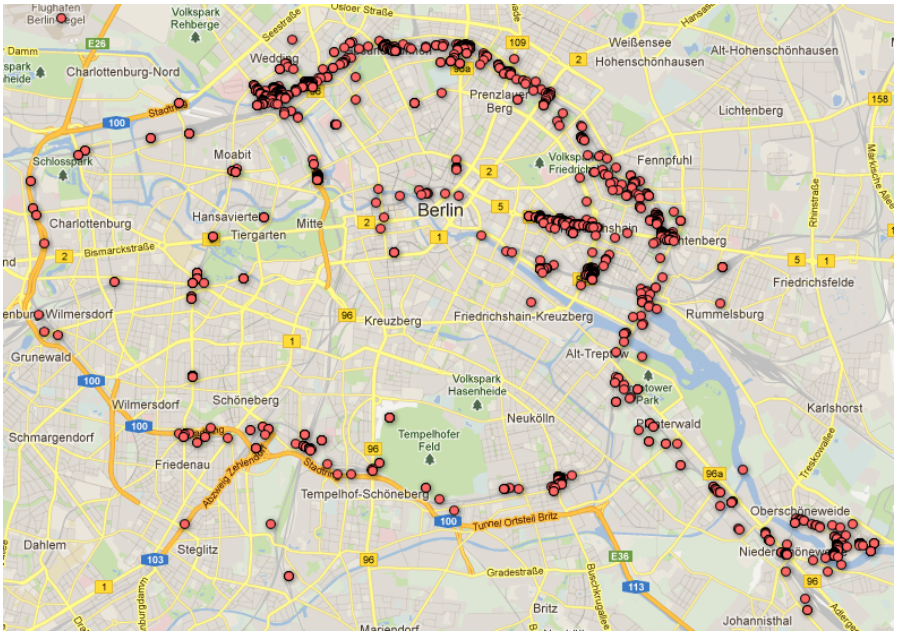


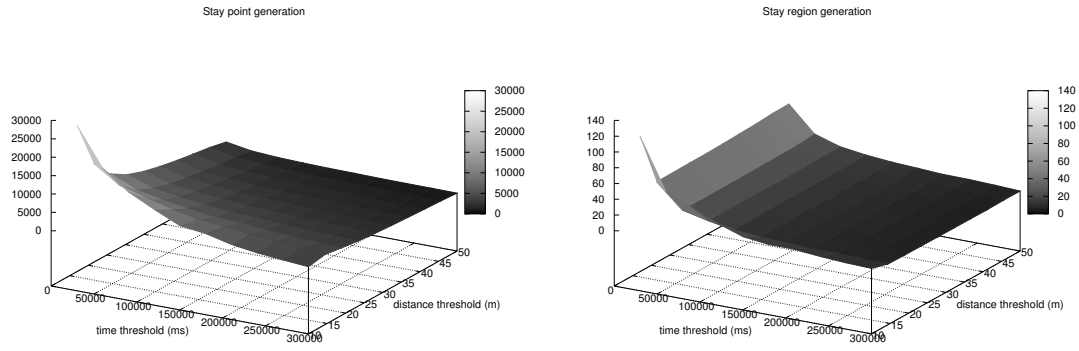
Figure 5.1: Locations of discovered devices from the captured data

Incidentally, the large arc from the home location to the lower right is directly on top of the public transportation railway.

First, some effects of different parameters for the clustering algorithm will be shown. Based on a subset of these parameters, the mobility prediction will be evaluated. Afterwards, the discovery scheme will be compared to eDiscovery (see section 2.3.2). All evaluations will be done with the data captured from the Android application, using simulation scripts that implement the algorithms and test them on the logged data.

5.1 Clustering and Mobility Prediction

To evaluate the generation of stay regions and stay points, the algorithm was tested with a set of parameters. Ideally, the stay regions depict a number of places that are actually significant, in that the user stayed in them for a significant amount of time. For the evaluation, the distance threshold for stay regions was set to $D_r = 100m$ and the minimal number of stay points in a region was set to $N = 10$. Higher values for the number of stay points in stay regions mostly lead to significantly less stay regions. The distance threshold for stay regions was always set to 100m, because this is a range that is mostly within the accuracy of network positioning systems, at least in urban areas.



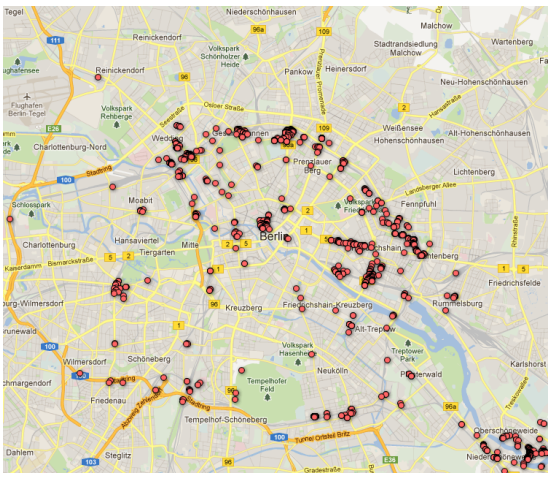
(a) Stay points

(b) Stay regions

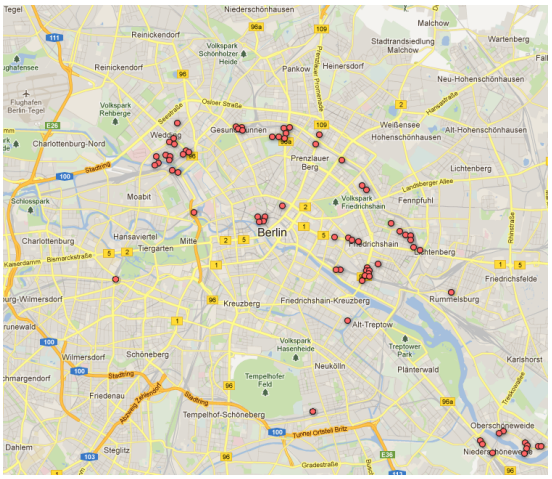
Figure 5.2: Number of generated stay points and stay regions for different thresholds

Using these basic static values, the stay point time thresholds were set to be between 30 and 300 seconds, in steps of 10 seconds. Additionally, the stay point distance threshold was set to be from 10 to 50 metres, in steps of 5 metres. Figure 5.2 contains the surface plots for stay point and stay region generation, based on these values. The curves show significant differences in stay point generation for the first few changes in the thresholds, with the differences gradually declining. The stay point generation seems to be affected fairly equally by differences in the time and distance threshold. Stay regions, on the other hand, seem to be largely unaffected by differences in stay region size after about 20 metres. The reason for this is most likely the exceptionally large number of stay points when using the lowest possible values. This indicates that the lowest values may be irrelevant, because staying within 10 metres for more than 30 seconds is extremely likely, and does not constitute any actual notability. Values for the distance threshold of stay points will be considered for up to 30 metres. Higher values seem irrelevant for the stay region generation. For the time thresholds, values of up to 120 seconds will be considered, because higher values reduce the number of stay regions too much. Higher values will also lead to later stay region generation, because the number of stay points grows more slowly.

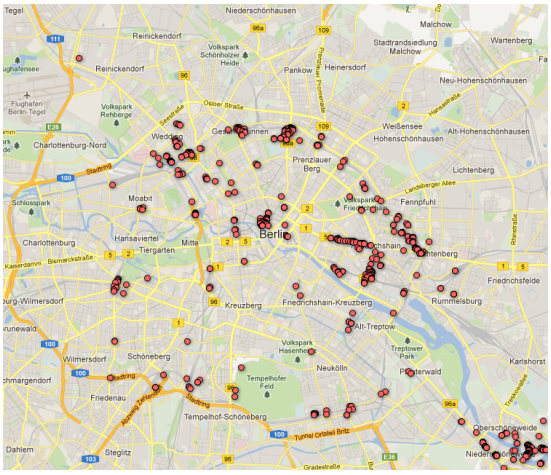
In Figure 5.3, the locations of generated stay points and stay regions are shown for two sets of parameters. Doubling the parameters to 60 seconds and 20 metres already affects the generation of stay regions significantly. As can be seen, stay regions tend to clump together in some areas, and using higher threshold values reduces this clumping, leading to more significant regions. The choice of clustering parameters also affects the mobility prediction. When the created stay regions are significant enough, the mobility



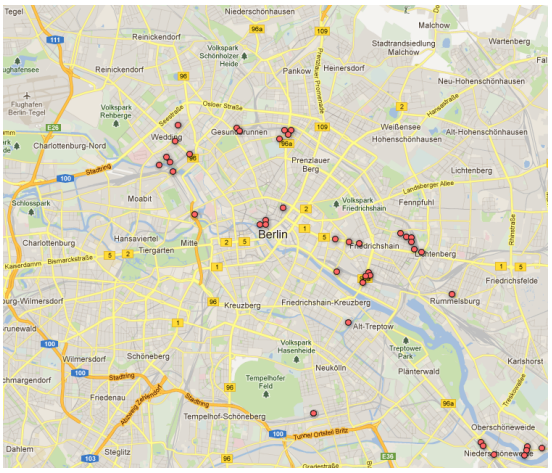
(a) Stay Points ($D_s = 10m$, $T_s = 30s$)



(b) Stay Regions ($D_s = 10m$, $T_s = 30s$, $D_r = 100m$, $N_r = 5$)



(c) Stay Points ($D_s = 20m$, $T_s = 60s$)



(d) Stay Regions ($D_s = 20m$, $T_s = 60s$, $D_r = 100m$, $N_r = 5$)

Figure 5.3: Locations of generated stay points and stay regions for $D_s = 10m$, $T_s = 30s$, $D_r = 100m$, $N_r = 5$ and $D_s = 20m$, $T_s = 60s$, $D_r = 100m$, $N_r = 5$

Clustering Parameters	Correct Predictions	Predictions	Accuracy
$D_s = 10m, T_s = 30s$	442	844	52,4%
$D_s = 20m, T_s = 60s$	398	701	56,8%
$D_s = 20m, T_s = 90s$	379	574	66,0%
$D_s = 20m, T_s = 120s$	381	567	66,1%
$D_s = 25m, T_s = 90s$	407	655	62,1%
$D_s = 30m, T_s = 120s$	377	577	65,3%

Table 5.2: Accuracy of mobility prediction for different clustering parameters

prediction will provide better results.

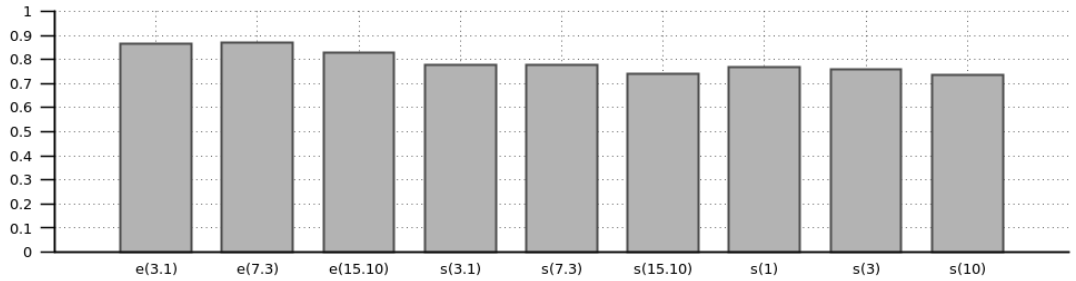
To evaluate the mobility prediction, it will be run parallel to the clustering algorithm on the captured data. The next probable location will be predicted as soon as the system detects that the user left a stay region. Based on the transitions stored for the stay region the user left, the next stay region likely to be visited next will be returned. When the user enters a stay region, it will be checked with the prediction. The result of the check is used to calculate the prediction accuracy as $A_p = \frac{No.of\ correct\ predictions}{No.of\ predictions}$. The prediction is tried with a subset of parameters for the clusters. As before, the distance threshold for stay regions is 100 metres, and the threshold for the number of stay points for each region is 10. Table 5.2 shows the 6 sets of parameters tested for the mobility prediction. As can be seen, the prediction accuracy is correlated to the generation of stay points. This is expected, because the clustering algorithm already identifies regions that have some significance by being visited by the user at least a number of times. The prediction accuracy directly influences the discovery scheme, because it relies on the transition information provided by the predictor. In the case of $D_s = 20m, T_s = 120s$, 66% of transitions were correctly identified, so in all of these cases, the discovery scheme could benefit from the prediction. The discovery evaluation in the following section will use this parameter set.

5.2 Discovery

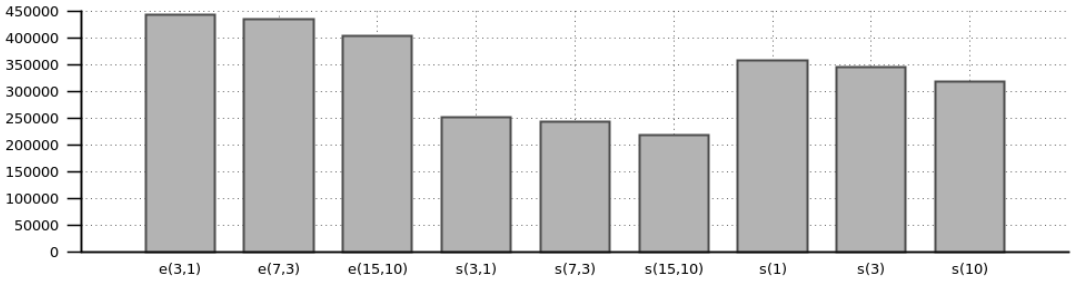
To evaluate the developed discovery scheme, it will be compared to eDiscovery, based on the data captured by the Android application, which used a constant probing scheme with an interval of 30 seconds and the standard inquiry window of 10.24 seconds. Using this dataset compare the discovery schemes with different parameters under the

same conditions. On the other hand, the dataset is fairly small and fairly specific, due to the fact that it was captured using only one phone on one person. Nevertheless, the comparison will provide some insights on the efficiency of the two algorithms. For eDiscovery, three different sets of values for N and I are chosen. The following evaluations will refer to these as *e(5,1)*, *e(7,3)* and *e(15,10)*. Additionally, the maximum inquiry interval will be set to 200 seconds for eDiscovery, which is the time it uses as presented in [HS12]. For the proposed scheme, the low value for the maximum interval will be 200 seconds, like in eDiscovery, and the high value will be 1800 seconds, like in STAR. To evaluate the impact of adapting the inquiry window, the proposed scheme will be evaluated with and without this adaptation. Accordingly, the values for N and I will be the same as for eDiscovery. The proposed scheme will be designated as *s(5,1)*, *s(7,3)* and *s(15,10)* for the schemes which adapt the discovery window, and *s(1)*, *s(3)* and *s(10)* for the schemes without adaptive discovery windows.

The captured data can not reproduce the actual contact times, the contacts will therefore be considered in range for 20 seconds before and after they have been discovered in the trace. The efficiency of each algorithm will be determined by the duration of inquiry and the number of contacts found from the trace. The duration of inquiry is the accumulated time in which the device is in inquiry mode. This time is directly correlated to the energy consumption of the scheme. Figure 5.4 plots the ratio of discovered contacts to the number of contacts contained in the dataset and the overall inquiry time for each of the 9 discovery schemes. The eDiscovery variants were tested 20 times each, while the proposed scheme was tested 10 times for each variant. This was necessary, because all schemes exhibit a random variable which may lead to skewed results. The plots show the average outcome for all runs. In terms of discovered contacts, the proposed scheme is consistently lower than the standard eDiscovery scheme. This effect can be attributed to low maximum inquiry interval of eDiscovery. The developed discovery scheme will sometimes experience long inquiry intervals, and thereby miss some contacts it did not expect. The difference in the contact ratio is between *e(3,1)* and *s(3,1)* is about 9%, while the inquiry time for *s(3,1)* is only 56% of the time used by *e(3,1)*. Based on the evaluation, the assumption that contact patterns can be predicted to improve the discovery scheme holds true. While the number of contacts discovered was smaller, the amount of energy saved is significant.



(a) Ratio of discovered contacts. eDiscovery exhibits the best ratio, with s(15,10) and s(10) discovering the least contacts



(b) Length of inquiry over the whole experiment. s(15,10) has nearly half the inquiry time of e(3,1)

Figure 5.4: Comparison of discovery schemes

6 Conclusions and Discussion

In this thesis, a discovery scheme was proposed that uses location information to reduce energy consumption. The scheme is aided by a clustering and a mobility prediction algorithm. The clustering is used to identify significant locations, while the mobility prediction is used to adapt to movement situations between locations. The scheme uses previously seen contacts at locations to adapt its maximum inquiry interval. By doing this, it effectively reduces the overall energy consumption of the node discovery process by having high delays in low-contact areas and low delays in areas with a high chance of contact. The scheme was adapted from eDiscovery[HHKM10], to which it was compared. The comparison, using a mobility and contact trace comprising about 3 months of data, showed that the developed scheme consumed up to 45% less energy, with only up to 15% less contacts discovered. This indicates the potential in adapting discovery schemes to locations. A downside to this is the added complexity of the algorithm, and the added computational requirements coming with it. eDiscovery is very simple and does not rely on information other than the number of peers discovered in the current probe. The developed scheme uses an incremental clustering algorithm and mobility prediction to facilitate the discovery. Even though the used algorithms are fairly simple, alternatives and optimizations are possible.

There are some potential optimizations to the clustering algorithm. Currently, when a new stay point is found, it will be compared to every unclustered stay point. These points may accumulate fast when a person exhibits fairly irregular mobility. In this case, many points may be scattered over the map, without enough points in close vicinity for them to be considered a stay region. Comparisons to these stay points could be optimized by dividing the map into a grid and automatically assigning any new stay point to the region it is located in. Any new point would then only have to be compared to points located in its own and any adjacent grid. On the other hand, the generation of stay points is directly correlated to the time threshold a user must have stayed at a place. This computation will therefore only occur every few minutes, on average. The computational complexity may still be overwhelming, especially on mobile phones. Another problem may arise in the fact that the number of stay regions is unbounded.

The mobility predictor may be affected by this, because it has to compute movement probabilities between a potentially large number of locations. Additionally, movement paths have to be recomputed each time a new stay region is generated. The recomputation also necessitates the construction of a new movement history. Further research on the mobility prediction and clustering could lead to improved algorithms.

A possible improvement to the mobility prediction would be the use of the SmartDC system described in [CTSC11], which uses adaptive duty cycling to sense location points and mobility. In contrast to the location sensing used in this thesis, SmartDC employs an adaptive multi-sensor approach, incorporating WiFi, GPS and network positioning. This enables localization with room-level accuracy while saving energy in relation to using, for instance, GPS only. While the network positioning used in this thesis does not consume a lot of power, its accuracy can be exceptionally low in certain circumstances. Additionally, the choice of using an O(1)-Markov predictor could be evaluated by comparing it to some other predictors. Feature-aided schemes like those that include time of departure may provide additional prediction accuracy.

Although the discovery schemes were compared using traces of real mobility, the schemes may behave differently in real experiments, where wireless interference and other factors may play a significant role. Nevertheless, the evaluation effectively reproduced the performance of eDiscovery measured in [HS12], which evaluated the algorithm in real settings. Further studies could aim to reproduce the results of this thesis in simulations, for instance in the ONE simulator[KOK09], and in real experiments.

Bibliography

[BEHP06] JA Burke, D Estrin, M Hansen, and A Parker, *Participatory sensing*, October (2006).

[CHC05] Augustin Chaintreau, Pan Hui, and Jon Crowcroft, *Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding*.

[Cis] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016 [Visual Networking Index (VNI)]*.

[CNC+08] Goutam Chakraborty, Kshirasagar Naik, Debasish Chakraborty, Norio Shiratori, and David Wei, *Analysis of the Bluetooth device discovery protocol*, Wireless Networks **16** (2008), no. 2, 421–436.

[CSTC12] Yohan Chon, Hyojeong Shin, Elmurod Talipov, and Hojung Cha, *Evaluating mobility models for temporal prediction with high-granularity mobility data*, 2012 IEEE International Conference on Pervasive Computing and Communications (2012), 206–212.

[CTSC11] Yohan Chon, Elmurod Talipov, Hyojeong Shin, and Hojung Cha, *Mobility prediction-based smartphone energy optimization for everyday location monitoring*, Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems - SenSys '11 (2011), 82.

[DARD07] Catalin Drula, Cristiana Amza, Franck Rousseau, and Andrzej Duda, *Adaptive energy conserving algorithms for neighbor discovery in opportunistic bluetooth networks*, Selected Areas in Communications, IEEE Journal on **25** (2007), no. 1, 96–107.

[EKKO08] Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott, *Working day movement model*, Proceeding of the 1st ACM SIGMOBILE workshop on Mobility models - MobilityModels '08 (2008), 33.

[Fal03] Kevin Fall, *A delay-tolerant network architecture for challenged internets*, Proceedings of the 2003 conference on Applications, technologies, architec-

tures, and protocols for computer communications - SIGCOMM '03 (2003), 27.

[HCS+05] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot, *Pocket switched networks and human mobility in conference environments*, Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking - WDTN '05 (2005), 244–251.

[HCY10] Pan Hui, Jon Crowcroft, and Eiko Yoneki, *Bubble rap: social-based forwarding in delay tolerant networks*, IEEE Transactions on Mobile Computing (2010).

[HHKM10] Bo Han, Pan Hui, VS Kumar, and MV Marathe, *Cellular traffic offloading through opportunistic communications: a case study*, Proceedings of the 5th (2010), 31–38.

[HS12] Bo Han and Aravind Srinivasan, *eDiscovery: Energy Efficient Device Discovery for Mobile Opportunistic Communications*, cs.umd.edu (2012).

[IL12] V Iyer and A Loukas, *Nest: A Practical Algorithm for Neighborhood Discovery in Dynamic Wireless Networks using Adaptive Beaconing*, es.ewi.tudelft.nl (2012).

[IMSR07] F. Ingelrest, N. Mitton, and D. Simplot-Ryl, *A Turnover based Adaptive HELLO Protocol for Mobile Ad Hoc and Sensor Networks*, 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (2007), 9–14.

[JL05] Ravi Jain and Dan Lelescu, *Model T: an empirical model for user registration patterns in a campus wireless LAN*, Mobile computing and networking (2005).

[KB10] Thomas Karagiannis and JY Le Boudec, *Power law and exponential decay of intercontact times between mobile devices*, Mobile Computing, IEEE (2010), no. March.

[KH05] D. Kotz and T. Henderson, *CRAWDAD: A Community Resource for Archiving Wireless Data at Dartmouth*, IEEE Pervasive Computing **4** (2005), no. 4, 12–14.

[KOK09] Ari Keränen, J. Ott, and T. Kärkkäinen, *The ONE Simulator for DTN Protocol Evaluation*, Proceedings of the Second International ICST Conference on Simulation Tools and Techniques (Gent, BELGIUM), Icst, 2009, p. 55.

[LKJ06] Dan Lelescu, UC Kozat, and Ravi Jain, *Model T++: an empirical joint space-time registration model*, Proceedings of the 7th (2006), 61–72.

[LM11] X Li and N Mitton, *Mobility prediction based neighborhood discovery in mobile ad hoc networks*, NETWORKING 2011 (2011).

[MM06] Mirco Musolesi and Cecilia Mascolo, *A community based mobility model for ad hoc network research*, Proceedings of the second international workshop on Multi-hop ad hoc networks: from theory to reality - REALMAN '06 (2006), 31.

[MSKY05] Kumiko Maeda, Kazuki Sato, Kazuki Konishi, and Akiko Yamasaki, *Getting urban pedestrian flow from simple observation: Realistic mobility generation in wireless network simulation*, simulation of wireless (2005).

[Mus09] Mirco Musolesi, *Mobility models for systems evaluation. A survey*, Middleware for Network Eccentric and Mobile Applications (2009), 1–28.

[MV03] Marvin McNett and Geoffrey M. Voelker, *Access and mobility of wireless PDA users*, ACM SIGMOBILE Mobile Computing and Communications Review **7** (2003), no. 4, 55.

[MW11] Christoph P. Mayer and Oliver P. Waldhorst, *Offloading infrastructure using Delay Tolerant Networks and assurance of delivery*, 2011 IFIP Wireless Days (WD) (2011), 1–7.

[New03] MEJ Newman, *Why social networks are different from other types of networks*, Arxiv preprint cond-mat/0305612 (2003).

[NM07] Anirudh Natarajan and Mehul Motani, *Understanding urban interactions from bluetooth phone contact traces*, Passive and Active Network (2007).

[SCP04] Jing Su, Alvin Chin, and Anna Popivanova, *User mobility for opportunistic ad-hoc networking*, and Applications, 2004 (2004), no. Wmcsa.

[SJ03] Nathan Srebro and Tommi Jaakkola, *Weighted low-rank approximations*, MACHINE LEARNING-INTERNATIONAL . . . (2003).

[SKJH06] Libo Song, David Kotz, Ravi Jain, and Xiaoning He, *Evaluating next-cell predictors with extensive Wi-Fi mobility data*, Mobile Computing, IEEE . . . (2006), 1–43.

[Tud05] Cristian Tuduce, *A Mobility Model based on WLAN Traces and its Validation*, 2005. 24th Annual Joint Conference of **00** (2005), no. c.

[VDN11] Long Vu, Quang Do, and Klara Nahrstedt, *Jyotish: A novel framework for constructing predictive model of people movement from joint Wifi/Bluetooth trace*, 2011 IEEE International Conference on Pervasive Computing and Communications (PerCom) (2011), 54–62.

[VNRG10] Long Vu, Klara Nahrstedt, Samuel Retika, and Indranil Gupta, *Joint bluetooth/wifi scanning framework for characterizing and leveraging people movement in university campus*, Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems, ACM, 2010, pp. 257–265.

[WAL11] John Whitbeck, M Amorim, and Yoann Lopez, *Relieving the wireless infrastructure: When opportunistic networks meet guaranteed delays*, Multimedia Networks (2011).

[WC02] Brad Williams and Tracy Camp, *Comparison of broadcasting techniques for mobile ad hoc networks*, Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing, ACM, 2002, pp. 194–205.

[WSM07] Wei Wang, Vikram Srinivasan, and Mehul Motani, *Adaptive contact probing mechanisms for delay tolerant applications*, Proceedings of the 13th annual ACM international conference on Mobile computing and networking - MobiCom '07 (2007), 230.

[ZL02] G.V. Zaruba and D. Levine, *Accelerated Neighbor Discovery in Bluetooth Based Personal Area Networks*, 2002.

[ZZXY10] Vincent W. Zheng, Yu Zheng, Xing Xie, and Qiang Yang, *Collaborative location and activity recommendations with GPS history data*, Proceedings of the 19th international conference on World wide web - WWW '10 (2010), 1029.

List of Algorithms

List of Figures

List of Tables

Eigenständigkeitserklärung

Hiermit versichere ich, Mathias Lenz, geboren am 15. Juni 1987, dass ich die vorliegende Masterarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Datum	Unterschrift