Technical Report

UCAM-CL-TR-713
ISSN 1476-2986

Number 713

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

People are the network: experimental design and evaluation of social-based forwarding algorithms

Pan Hui

March 2008

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

© 2008 Pan Hui

This technical report is based on a dissertation submitted July 2007 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Churchill College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

People are the network: experimental design and evaluation of social-based forwarding algorithms

Pan Hui

Summary

Cooperation binds but also divides human society into communities. Members of the same community interact with each other preferentially. There is structure in human society. Within society and its communities, individuals have varying popularity. Some people are more popular and interact with more people than others; we may call them hubs. I develop methods to extract this kind of social information from experimental traces and use it to choose the next hop forwarders in Pocket Switched Networks (PSNs). I find that by incorporating social information, forwarding efficiency can be significantly improved. For practical reasons, I also develop distributed algorithms for inferring communities.

Forwarding in Delay Tolerant Networks (DTNs), or more particularly PSNs, is a challenging problem since human mobility is usually difficult to predict. In this thesis, I aim to tackle this problem using an experimental approach by studying real human mobility. I perform six mobility experiments in different environments. The resultant experimental datasets are valuable for the research community. By analysing the experimental data, I find out that the inter-contact time of humans follows a power-law distribution with coefficient smaller than 1 (over the range of 10 minutes to 1 day). I study the limits of "oblivious" forwarding in the experimental environment and also the impact of the power-law coefficient on message delivery.

In order to study social-based forwarding, I develop methods to infer human communities from the data and use these in the study of social-aware forwarding. I propose several social-aware forwarding schemes and evaluate them on different datasets. I find out that by combining community and centrality information, forwarding efficiency can be significantly improved, and I call this scheme BUBBLE forwarding with the analogy that each community is a BUBBLE with big bubbles containing smaller bubbles. For practical deployment of these algorithms, I propose distributed community detection schemes, and also propose methods to approximate node centrality in the system.

Besides the forwarding study, I also propose a layerless data-centric architecture for the PSN scenario to address the problem with the status quo in communication (e.g. an infrastructure-dependent and synchronous API), which brings PSN one step closer to real-world deployment.

Acknowledgments

I would like to thank my supervisor Prof. Jon Crowcroft for endless jolly discussions and beers in the Castle. I would also like to thank Dr Christophe Diot, Dr James Scott and Dr Augustin Chaintreau for their wonderful supervision and help during the first one and a half year of my PhD; and Dr Eiko Yoneki for the very enjoyable collaboration in my third year. I would like to thank Mr Shu-Yan Chan for his help with some programming.

I want to thank to Dr Steven Hand, Dr Pietro Lio, Dr Andrew Moore, Dr Mirco Musolesi, Dr Sid Chau, Mr Derek Murray, Miss Meeyoung Cha, Dr Andrea Passarella, Dr Andrew Warfield, Dr Meng-How Lim, Mr Hamed Haddadi and many others for their insightful comments on the dissertation and related papers; and Dr Ralph Kling and Mr Vincent Hummel from Intel for their help on the iMote platform. I also want to thank the people who have shared the same office with me and the friends I made during my internship in Intel Cambridge and Thomson Paris. I have had a very good time over these three years.

Finally I want to thank my parents for their endless support, the Sir Edward Youde Foundation for the main financial support of my PhD program, and Intel Research Cambridge and Thomson Rsearch Paris for the interesting working experiences.

List of Publications

[10-2007] Eiko Yoneki, Pan Hui, Shu-yan Chan and Jon Crowcroft. A Socio-Aware Overlay for Multi-Point Asynchronous Communication in Delay Tolerant Networks. In ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), October 2007, Crete Island, Greece.

[9-2007] Eiko Yoneki, Pan Hui, and Jon Crowcroft. Visualizing Community Detection in Opportunistic Network. In ACM MobiCom 2007 Workshop on Challenged Networks (CHANTS), September 2007, Montreal, Canada.

[09-2007] Jing Su, James Scott, Pan Hui, Eben Upton, Meng How Lim, Christophe Diot, Jon Crowcroft, Ashvin, Goel, and Eyal de Lara . Haggle: Clean-slate networking for mobile devices. In Conference on Ubiquitous Computing (Ubicomp), September 2007, Innsbuck, Austria.

[08-2007] Pan Hui, Eiko Yoneki, Shu-yan Chan and Jon Crowcroft. Distributed Community Detection in Delay Tolerant Networks. In Sigcomm Workshop MobiArch ’07, August, Kyoto, Japan.

[06-2007] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of Human Mobility on Opportunistic Forwarding Algorithms. IEEE Transactions on Mobile Computing, Volume 6, Issue 6 (June 2007).

[05-2007] Pan Hui and Jon Crowcroft. Bubble Rap: Forwarding in small world DTNs in ever decreasing circles. University of Cambridge Computer Laboratory Technical Reports, UCAM-CL-TR-684, May 2007.

[03-2007] Pan Hui and Jon Crowcroft. How Small Labels create Big Improvements. International Workshop on Intermittently Connected Mobile Ad hoc Networks (ICMAN). March 19-23, 2007, White Plains, NY.

[12-2006] Pan Hui and Jon Crowcroft. How Small Labels create Big Improvements. Second coNEXT Conference Student Workshop Poster, December 4-7, 2006,Lisboa, Portugual.

[10-2006] Pan Hui, Jeremie Leguay, Jon Crowcroft, James Scott, Timur Friedman and Vania Conan. Osmosis in Pocket Switched Networks. First International Conference on Communications and Networking in China (ChinaCom). Oct 25-27, 2007, Beijing, China.

[04-2006] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, James Scott. Impact of Human Mobility on the Design of Opportunistic Forwarding Algorithms. In IEEE INFOCOM 2006,April 23-29, Barcelona, Spain.

[01-2006] James Scott, Pan Hui, Jon Crowcroft, and Christophe Diot. Haggle: A Networking Architecture Designed Around Mobile Users. In IFIP WONS 2006, January 18-20, Les Menuires, France.

[10-2005] Pan Hui, Augustin Chaintreau, Richard Gass, James Scott, Jon Crowcroft, and Christophe Diot. Pocket Switched Networking: Challenges, Feasibility, and Implementation Issues. In Workshop on Autonomic Communication (WAC), Oct 3-5, 2005, Vouliagmeni-Athens, Greece.

[09-2005] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket Switched Networks and the Consequences of Human Mobility in Conference Environments. In SIGCOMM Workshop on Delay Tolerant Networking, September, 2005, Philadelphia, USA.

[02-2005] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, James Scott. Pocket Switched Networks: Real-world mobility and its consequences for opportunistic forwarding. University of Cambridge, Computer Lab. Technical Report UCAM-CL-TR-617. February 2005

List of Acronyms

ADU:	Application-level Data Unit
AP:	Access Point
API:	Application Programming Interface
CCDF:	Cumulative Distribution Function
DO:	Data Object
DTN:	Delay Tolerant Network/Networking
FO:	Forwarding Object
GPL:	General Public License
GPRS:	General Packet Radio Service
GSM:	Global System for Mobile communications
HTTP:	Hypertext Transfer Protocol
IP:	Internet Protocol
INS:	Intentional Naming System
MAC:	Media Access Control
MANET:	Mobile Ad Hoc Networks
MCP:	Multiple-Copy-Multiple-Hop
NO:	Name Object
POP:	Post Office Protocol
PSN:	Pocket Switched Network/Networking
SMTP:	Simple Mail Transfer Protocol
SNMP:	Simple Network Management Protocol
SQL:	Structured Query Language
TTL:	Time to Live
WNA:	Weighted Network Analysis

Contents

Chapter 1

Introduction

This dissertation is concerned with the design and evaluation of forwarding algorithms in Pocket Switched Networks (PSNs), which are a type of Delay Tolerant Network (DTN) targeting mobile humans [Fal03]. A PSN uses contact opportunities to allow humans to communicate without network infrastructure. I show that, by adding social information such as community and centrality, the forwarding efficiency of a PSN can be significantly improved over stateless "oblivious" forwarding schemes and also state of the art encounters predicting algorithm. The key contribution of this thesis is the design of efficient (in term of delivery ratio and cost) social-based forwarding algorithms from the empirical understanding of human mobility, interaction and social structures.

Forwarding is a challenging problem in PSN. Unlike traditional Internet and Mobile Ad Hoc Network (MANET), an end-to-end path for each node pair is usually impossible in PSN because either the network topology is changing too fast or the network is too sparse for full connectivity. It means that the traditional routing table strategy is not applicable to solve forwarding problem in this kind of environments. We require an efficient data forwarding mechanism that copes with dynamical, repeated disconnection and re-wiring. Some state of art routing algorithms [JLW05] [LDS04] in this area still provide forwarding by building and updating routing tables whenever mobility occurs. I believe this approach is not cost effective for a PSN, since mobility is often unpredictable, and topology changes can be rapid. Rather than exchange much control traffic to create unreliable routing structures, I prefer to search for some characteristics of the network which are more tolerant to mobility. A PSN is formed by people. Those people's social relationships may vary much more slowly than the topology, and therefore can be used for better forwarding decisions. Furthermore, if we can detect these social mobility patterns online in a decentralised way, we can put the algorithms into practical applications.

I propose the BUBBLE algorithm, with the intention of bringing in a concise concept of *community* into PSN forwarding to achieve significant improvement of forwarding efficiency. BUBBLE combines the knowledge of community structure with the knowledge of node *centrality* to make forwarding decisions. There are two intuitions behind this algorithm. Firstly, people have varying roles and popularities in society, and these should be true also in the network – the first part of the forwarding strategy is to forward messages to nodes which are more popular than the current node. Secondly, people form communities in their social lives, and this should also be observed in the network layer – hence the second part of the forwarding strategy is to identify the members of destination communities, and to use them as relays.

In this chapter I outline the background issues that motivated this work, and state the contributions that are described in this dissertation. After that I summarise the contents of each chapter.

1.1 Metrics for Evaluating Forwarding Efficiency

The aim is to show that BUBBLE and its sub-class of algorithms are efficient forwarding algorithms in term of delivery ratio and cost. Two forwarding algorithms with similar delivery ratio, the one with lower delivery cost is classified as more efficient and vice versa for the delivery cost case.

For all the emulations/simulation conducted to compare forwarding efficiency in this thesis, I have the following two metrics, and for these two metrics, I compute the 95th percentile using t-distribution.

Delivery Ratio: The proportion of messages that have been delivered out of the total unique messages created.
Delivery Cost: The total number of messages (include duplicates) transmitted across the air. To normalize this, I divide it by the total number of unique messages created.

For some cases, I also compute the Hop-count-distribution for the deliveries, which is the distribution of the number of hops needed for all the deliveries, and which reveals the social distance between sources and destinations.

In this thesis, I am not going to do end-to-end evaluation of applications using these algorithms. I only propose a simple unicast communication scenario with delay tolerant messaging service. I will limit my evaluation to delivery ratio and delivery cost in a network where all connectivity is short-range wireless (i.e., no wired backhaul allowed). Although the PSN concept allows using infrastructures (e.g. wireless access point) to tunnel the messages, I will not include it in this thesis.

1.2 Social Networks

Society naturally divides into *communities* according to needs for cooperation or selection. In sociology, the idea of *correlated interaction* is that an organism of a given type is more likely to interact with another organism of a same type than with a randomly chosen member of the population [Oka05]. If the correlated interaction concept applies, then our intuition is that using community information to influence forwarding paths may be advantageous.

The PSN forwarding problem is then turned into a well known community detection problem [NG04] [New06] [New04b] [DDDGA05] [Cla05]. Besides community, *familiar strangers* is another useful relationship which can be useful for data delivery. Studied by Milgram [Mil77], familiar strangers are people you meet regularly but do not spend time with, such as people who take the same bus or underground as you every day to the work, or people use the same laundry room. This kind of regularity can be useful for predicting encounter, for example we can usually predict the arrival of buses because of their regular schedule. I believe that by observing the contact patterns of the mobile devices, we can infer this social information.

Understanding a network and a node's participation in the network is important. Centrality measurements give insight into the roles and tasks of nodes in a network. Freeman [Fre77] defined several centrality metrics to measure the importance of a node to the network. Betweenness centrality measures the number of times a node falls on the shortest path between two other nodes. This concept is also valid in a temporal network, which is a type of network with time-dependent connectivity [KKK02]. In a PSN, it can represent the importance of a node for relaying traffic for others in the system. To determine the overall centrality of a vertex, p_k, we need merely to sum its partial betweenness values for all unordered pairs of points where $i \neq j \neq k$:

$$C_B(p_k) = \sum_{n} \sum_{n} b_{ij}(p_k), i < j \tag{1.1}$$

where n is the number of points in the graph. The sum, $C_B(p_k)$, is an index of the overall partial betweenness of point p_k. Whenever p_k falls on the only geodesic path connecting a pair of points, i and j, $C_B(p_k)$ is increased by 1 ($b_{ij}(p_k) = 1$). When there are alternative geodesic paths, $C_B(p_k)$ is increased in proportional to the frequency of occurrence of p_k among those alternatives. Closeness centrality yields the node with the shortest path to all others and the best visibility in a graph of relationships. It is a measure of how long it will take data to spread to the others in this graph. The closeness $C_C(a)$ for a vertex a is the inverse sum of distances (i.e. hop count) to other nodes b:

$$C_C(a) = \ 1 \ / \ \sum_{b} d_{ab} \tag{1.2}$$

In this dissertation, I will concentrate on using the above social network concepts to help us to choose a good data carrier for a specific destination.

1.3 Pocket Switched Networks

PSNs are a type of DTN, targeted at mobile users. Mobile workers move between connectivity islands (e.g. WiFi at home and work). Outside these islands, end-to-end connectivity becomes expensive, slow, or simply unavailable. PSN is the new networking paradigm, which uses human mobility and store-and-forward strategy to solve the communication problem outside the connectivity islands. On the other hand, there is a huge amount of untapped resources in portable networked devices such as laptops, PDAs and mobile phones, including local wireless bandwidth (e.g. 802.11 and Bluetooth), storage capacity, CPU power, and multimedia data. These resources should be utilised. Furthermore, the communication between users is not always necessarily to pass through the Internet. According to a questionnaire survey amount 70 participants in the Computer Laboratory University of Cambridge, around 50% of their email exchanges are among people they met daily. Another motivation is that the information provide by the Internet may not best satisfy the interest of the local users, for example an user may be more interested in a video clip of his friend, Britney Spears, instead of the MTV of the singer Britney Spears that is usually what Google search will return to you. Empirical result about social search are also observed by other researchers [MGD06]. In this aspect, PSN unleashes the power of local, social and community search and communication.

Currently the only scarce resource is battery power, but advances in power engineering and battery technologies have meant that mobile phones now last for a week on a single charge, while remaining in constant network contact (although in idle mode). I expect that this innovation will continue, allowing devices to participate in wireless networks while minimizing power consumption. PSN focus on multi-hop delivery and data searching in neighbour's cache. I 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. This is the goal of PSN. To further illustrate the PSN scenario, I give two motivating examples here.

First, let us consider a Japanese tourist, Nami, in Paris. Nami uses her HTC Touch mobile phone to take pictures of Paris. She is very excited when she arrives the Eiffel tower, and she wants to take pictures of the beautiful tower from different angles.[1] But her mobile phone is running out of storage because in the same day she already took 300 pictures. Nami can delete some pictures, but she feels pity since every picture is so nice and unique to her. She wants to send some pictures to her friend, Eiko, who is also visiting Paris, so Eiko can store the pictures for her temporarily. Nami cannot find an Internet Cafe around Eiffel tower and it is too expensive to send via GPRS, especially using her roaming phone, and she is leaving Paris tomorrow. Nami is supposed to be able to send her pictures to Eiko or even to her own email account by multi-hop delivery using short-range radio, but the current technologies disappoint this young lady.

A second motivating example is set on a train going towards London. Charlie wants to find out about restaurants in London using his laptop. He does not have any GPRS connection (and may not be willing to pay for it if he did, or may be out of coverage range, or the train might actually be a plane). Currently, Charlie would not bother even trying to perform this task, as he probably knows that his web browser (which is the obvious application to end users for obtaining information) only works when he has a connection to a wireless access point (AP). In this case, the frustrating thing is that the data is highly likely to be present on many other devices within wireless range of Charlie, since others going to London may well have looked

[1] Yes, she is Japanese! What can we say?

up restaurants before they departed, or on the train (if they did have GPRS access etc). However, with the existing architecture, that information is not available to Charlie.

There are also PSN related prior systems that use a wireless LAN when wide-area connectivity is not available. Bayou (from PARC in 1995) maintained a calendaring application where wireless devices opportunistically synchronize with each other, and where connectivity to an infrastructure server was only intermittently [TTP+95]. More recently, Pastwatch is a peer-to-peer CVS-like system from MIT that allows a group of nodes to become disconnected from the "main" repository, and to form a replicated read-write source repository as a local cluster while partitioned from the main server [YCM06]. Although Pastwatch does not do automatic discovery and name resolution of nearby nodes, it is still a good example of an application that tolerates wide-area disconnection (e.g. PSN scenario) with a high level of functionality remaining. These prior works also strengthen our motivations of PSN research. But because this thesis is focus on forwarding, I will leave more discussion about applications and systems in the Appendix and also as future work.

1.4 Haggle Architecture

Haggle architecture [2] is a ground-up redesign of networking for mobile devices to support the mobile user scenario, more particular the PSN scenario. The key idea behind Haggle is to have a data-centric architecture [ASS+] where applications do not have to concern themselves with the mechanisms of transporting data to the right place, since that is what has made them infrastructure-dependent. By delegating to Haggle the task of propagating data, applications can automatically take advantage of any connection opportunities that arise, both local neighborhoods opportunities and connectivity with servers on the Internet when available. I identify four design decisions for Haggle that follow on from this.

1. **Data Persists inside Haggle**: The data on each node in Haggle must be visible to and searchable by other nodes (with appropriate security/access restrictions applied). This facilitates operation of our motivating web example, in that the public webpage needed by one person can be found despite it being in another person's device. In practice, this means that Haggle must manage persistent data storage for applications, instead of applications storing data in a separate file system.
2. **Networking Protocols inside Haggle**: Any application-layer networking protocol includes implied assumptions about the type of network available. For example, client-server protocols such as SMTP, POP and HTTP assume that Internet-based servers are contactable. With Haggle, I place networking protocol support inside Haggle itself, allowing me to present a data-centric rather than connection-centric abstraction to applications.

[2]Haggle has two meanings, the first meaning is "ad hoc Google", and the second meaning represents the haggling scenario of finding data carriers.

3. **Name Graphs supporting Late Binding**: Haggle maintains its own naming repository (it obviously cannot rely on remote look-up of this data), with mappings from user-level names to protocol-specific names specifying the various ways to get to the user-level name. Furthermore, the whole set of mappings (the “name graph”) is transmitted along with the data, allowing even intermediate (i.e. non-source) nodes to bind to protocol-specific names as late as possible [AWSBL].
4. **Centralised Resource Management**: One role of the networking architecture on every device is to decide what to do with each of its network interfaces *now*. Haggle contains a centralised resource management component, which decides on a cost/benefit comparison basis what tasks it chooses to perform on each network interface at a given moment.

The details of the Haggle architecture design can be found in the Appendix.

1.5 Delay Tolerant Networks

The existing TCP/IP based Internet operates on the principle of providing end-to-end inter-process communication using a concatenation of potentially dissimilar link-layer technologies. The standardisation of the IP protocol and its mapping into network-specific link-layer data frames at each router as required supports interoperability using a packet-switched model of service. Although often not explicitly stated, a number of key assumptions are made regarding the overall performance characteristics of the underlying links in order to achieve smooth operation: an end-to-end path exists between a data source and its peer(s), the maximum round-trip time between any node pair in the network is not excessive, and the end-to-end packet drop probability is small. A class of so-called *challenged networks*, which may violate one or more of these assumptions, is becoming important and may not be well served by the current end-to-end TCP/IP model. Challenged networks arise primarily as a result of various forms of host and router mobility, but may also come into being as a result of disconnection due to power management or interference. Examples of such networks include Terrestrial Mobile Networks, Exotic Media Networks, Military Ad-Hoc Networks, and Sensor and Sensor/Actuator Networks. The architecture for Delay Tolerant Networking (DTN) [Fal03] seeks to address the communication needs of these *challenged* environments. This architecture proposes a message based store-and-forward overlay network that leverages a set of convergence layers to adapt to a wide variety of underlying transports. In addition, the model also espouses novel approaches to application structuring and programming interface, fragmentation, reliability, and persistent state management.

Both PSN and DTN are designed to solve intermittent connection problem. DTN focuses more on environments with more predictable mobility (e.g. satellites, buses), and PSN targets on mobile humans, which have more difficult predictability. As I mentioned in the previous section, PSN is not only to solve connectivity problem, but also with the insight of local communication, community sharing, and social search from neighbours' and neighbours' neighbours' caches. From the architecture point of view, the Haggle architecture is a data-centric clean slate design for PSN, where applications do not have to concern themselves with the mechanisms of transporting data to the right place, since that is what has made them infrastructure-dependent. The data-centric principle of Haggle is that the data on each node in Haggle must be visible to and searchable by other nodes (with appropriate security/access restrictions applied). In other words, relationships between application data units (e.g. a webpage and its embedded images) should be representable in Haggle, and applications should be able to search both locally and remotely for data objects matching particular useful characteristics.

We can see that we need a completely new paradigm to consider forwarding in this new communication model. In this thesis, I look at two human social structures, community and centrality, which are very important for the data-centric forwarding.

1.6 Forwarding in PSNs

Forwarding in PSNs is a challenging problem since human mobility is usually unscheduled and difficult to predict. Quite a lot of work has been done on forwarding in DTNs and the related mobile ad hoc networks in the literature. Vahdat *et al.* proposed the epidemic routing [VB00] which is similar to the "oblivious" flooding scheme I evaluated in this thesis. Spray and Wait [SPR05] is another "oblivious" flooding scheme but with a self-limited number of copies. Grossglauser *et al.* proposed the two-hop relay scheme [GT02] to improve the capacity of dense ad hoc networks.

Many approaches calculate the probability of delivery to the destination node, where the metrics are derived from the history of node contacts, spatial information and so forth. The pattern-based Mobyspace Routing by Leguay *et al.* [LFC06], location-based routing by Lebrun *et al.* [Leb05], context-based forwarding by Musolesi *et al.* [MHM05] and PROPHET Routing [LDS04] fall into this category. PROPHET uses past encounters to predict the probability of future encounters. The transitive nature of encounters is exploited, where indirectly encountering the destination node is evaluated. Message Ferry by Zhao *et al.* [ZAZ04] takes a different approach by controlling the movement of each node. Recent attempts to uncover a hidden stable network structure in DTNs such as social networks have emerged. For example, SimBet Routing [DH07] uses ego-centric centrality and its social similarity. Messages are forwarded towards the node with higher centrality to increase the possibility of finding the potential carrier to the final destination. RANK algorithm introduced in this thesis uses betweenness centrality in a similar manner to SimBet routing. On the other hand, BUBBLE exploits further community structures and combines it with RANK for further improvement of forwarding efficiency. The mobility-assisted Island Hopping forwarding [NSDG06] uses network partitions that arise due to the distribution of nodes in space. Their clustering approach is based on the significant locations for the nodes and not for clustering nodes themselves. Clustering nodes is a complex task to understand the network structure for aid of forwarding.

Finally, I emphasise that I take an experimental rather than theoretical approach, which makes a further difference from the other work described above. I believe that the proper way to understand forwarding is first to understand human mobility and interaction patterns, otherwise the algorithms designed may be mathematically beautiful but far from reality. We need to know how the people move, how they meet each other, how they interact with each other, and whether they move as a group. Can we find out regular or long-term patterns? This way of reasoning turns forwarding in PSNs into a social network problem. I look at what kind of social information is useful for forwarding, and how can this be extracted efficiently from daily contact patterns.

1.7 Security and Privacy

In this thesis, security and privacy have not been addressed as key concerns; I chose to narrow the scope of the problem to exclude them, so as to allow me to make progress. I intend to introduce security primitives as a core concern in the future PSN research. However, I have made an initial analysis of the potential security threats that PSN and the Haggle architecture raise, discussed below.

Many data security issues in PSN can be handled using standard security techniques such as encryption, access control, and data signing. PSN makes it more likely that there will be a man-in-the-middle attack. One proviso is that many security techniques rely on access to a trusted third party, e.g. a certificate signing authority. This access may be available less often when using PSN. One interesting approach would be to accept data which is uncheckably signed, but somehow mark it (both internally and to the user) as “untrusted” until the signature can be checked through infrastructure access.

There are particular security and privacy issues in the use of name graphs in the Haggle architecture. A name graph can contain sensitive information, e.g. a user’s email address and/or phone number, or the number and type of a user’s devices (and hence how worthwhile it is to rob the user). Since Haggle potentially exposes the full graph to everyone who can see an FO with the graph, this could prove to be a breach of privacy. One solution might be to restrict trust to particular groups of users, e.g. the personnel of a company, and avoid sending messages through untrusted nodes, except when the name graph and data have been encrypted and authenticated to the extent that those nodes could not obtain any useful information, and could only help by passing the data on to non-privacy-sensitive names (e.g. MAC addresses).

There are also privacy issues to do with neighbor discovery protocols, since one’s devices essentially beacon their identity. This could allow tracking of the user. This problem is not unique to Haggle, and many devices already essentially act as beacons, e.g. laptops using 802.11 placing their MAC addresses on each frame.

Resource theft or resource denial-of-service is an interesting issue for any system in which user-owned nodes cooperate to achieve their goals, and resources are limited. In Haggle, we have a built-in mechanism to cope with this, namely the Resource Manager, which already makes judgments taking into account the utility of a given action to the user, and the device owner's preferences. On the other hand, this offers a single point of attack whereby a remote exploit might allow an attacker to take full control of the device, so securing the Resource Manager will be of particular concern.

Finally, we might ask the question of what motivates any node to spend its resources assisting any other node. An incentive to cooperate can be created in many ways — using reputation systems, micro-payments, or social kudos/disapproval. Furthermore, in some possible deployments of Haggle, e.g. within an enterprise, there is a pre-existing incentive to cooperate so this may not be a problem.

1.8 Contributions

My first contribution is the conducting of several real human mobility experiments which provide valuable datasets for the research community. I programmed the sensors (known as iMotes, which run TinyOS and are equipped with Bluetooth) to log other Bluetooth devices within communication range. I have conducted 6 experiments in different environments including conferences, research labs, a university town, and a metropolitan city, with up to 80 participants. For some experiments, I also deployed fixed iMotes at some city hot spots and key areas in a conference to provide approximate location information and mimic infrastructure. I analysed the human contact time and inter-contact time distributions and discovered that the inter-contact time for each pair follows a heavy-tail distribution over the range of 10 minutes to 1 day. I analytically studied the impact of the power-law coefficient on PSN forwarding and also empirically analysed the "oblivious" forwarding schemes.

A further contribution is the apply of the community detection algorithms from complex network study to the mobility trace analysis, and the propose and evaluation of three distributed community detection algorithms for mobile devices. Most of the mobility traces available and mobility experiments to date have no *a priori* community information, which makes study of social-aware or community based forwarding impossible [EP06] [MV05] [CHC$^+$06] [HCS$^+$05] [HKA04]. Community detection has been well studied but is still a popular problem in the complex networks and bio-informatics communities. It has been used to analyse protein structure, human social relationships, and internet AS-level clustering. In this thesis, I apply and adapt two community detection algorithms named weighted network analysis (WNA) and K-CLIQUE detection to infer human communities from mobility traces, which are further used for the forwarding study. In real deployment, we do not expect a central server to collect all the traces from the mobile devices for community detection, so I also propose three schemes for distributed community detection of mobile devices. I evaluate these distributed algorithms against the centralised methods, and find the results to be quite satisfactory.

The final main contribution is the design and evaluation of several social-aware forwarding algorithms using community and centrality information. I propose LABEL which makes use of *a priori* social information, RANK which makes use of pre-calculated betweenness centrality, and BUBBLE which makes use of centrality to move the messages away from the source and use community information to identify the destination group. I evaluate the algorithms against flooding, the "oblivious" multiple-hop-multiple-copy (MCP) scheme (controlled flooding scheme by limiting the number of hops and number of copies), and PROPHET [LDS04]. I find out that by combining community and centrality information, we can achieve a delivery ratio close to controlled flooding and PROPHET but with much less cost. As a by-product of these algorithms, I also uncover several properties which are important for a new human interaction/mobility design.

I have an additional contribution of the design of Haggle architecture, a clean-slate data-centric architecture around mobile users, which is considered to be a side-track work from the forwarding algorithm design so I include it only in the Appendix.

1.9 Outline

The remainder of this dissertation is structured as follows.

In Chapter 2 I describe the iMote experiments I have conducted and also the datasets I obtained from the community. I analyse the inter-contact distribution for all the datasets and also the impact of the distribution on forwarding strategies. A section is also dedicated to the empirical analysis of "oblivious" forwarding on the traces.

In Chapter 3 I infer the human community structures from the mobility traces using WNA and K-CLIQUE. I also propose three distributed community detection algorithms for mobile devices.

In Chapter 4 I introduce the simple social-aware forwarding algorithm called LABEL, which is evaluated on a dataset with *a priori* community information.

In Chapter 5 I present the BUBBLE algorithm which is the social-aware algorithm making use of both community and centrality information.

In Chapter 6 I conclude the thesis with a discussion of ongoing and future work.

In the Appendix I present the clean-slate data-centric Haggle architecture, the problem with the status quo, the design principles, and the detailed design.

Chapter 2

Measuring Human Mobility

In Pocket Switched Networks (PSNs), mobility determines the communication opportunities when access infrastructure is not available. Studying human mobility can help us understand the constraints of opportunistic communication and to design practical and effective forwarding strategies. Killer applications and security measures can also be inferred from the human mobility and interaction pattern [1], however, they are not the main focus of this thesis so I will not go into the details. I will limit my evaluation to multi-hop delivery, assuming a messaging service.

This chapter concentrates on several peer-contact-based human mobility experiments I have conducted during this thesis period, and the analysis of the human mobility patterns from these traces. To ensure generality of the analysis results, I also include mobility traces from other experiments, i.e. WiFi access point logs from Dartmouth College [HKA04]. I need to emphasise here that while previous works looked at wireless network proximity in mobile animals [JOW$^+$02a, SH03], mine is a pioneering study of human proximity using mobile devices, and analysis of the impact of human contact patterns for opportunistic communications.

The chapter is the result of collaboration with my supervisor Prof. Jon Crowcroft, and also Dr. Augustin Chaintreau, Dr. Christophe Diot, Dr. James Scott, and Richard Gass. Statistical analysis of human contact and inter-contact patterns and mathematical study of the impact of inter-contact distribution on forwarding are the contribution of Dr. Chaintreau; my contributions are mainly to the experimental platforms, experimental deployment, and the inference of the human social patterns from the traces. Most of the text and results in this chapter are extracted or summarised from a WDTN workshop paper [HCS$^+$05] with me as the first author and an IEEE Transactions on Mobile Computing paper [CHC$^+$07] with Dr. Chaintreau as the first author and me as the second author. Vincent Hummel and Dr. Ralph Kling from Intel provided very important support for the iMote platform and software development on the iMotes.

[1]For example, city wide alternative reality gaming applications can plan where to put their infrastructures and distribute hints if they know how the people move; community-based data sharing and caching applications can plan their caching strategies if they know the community structures.

2.1 Introduction

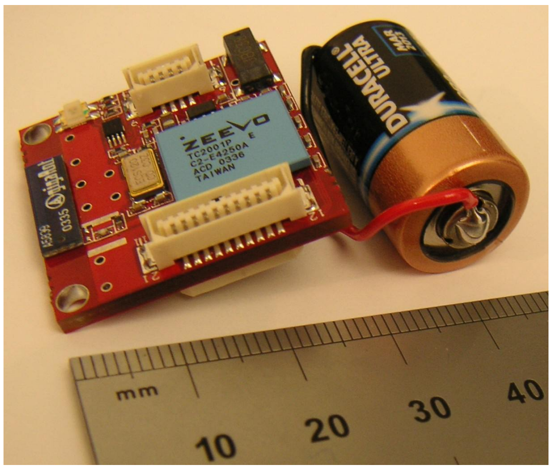
The increasing popularity of devices equipped with wireless network interfaces (such as cell phones or PDAs) offers new communication services opportunities. Such mobile devices can transfer data in two ways - transmitting over a wireless (or wired) network interface, and carrying from location to location by their users (while stored in the device). They can therefore participate in what has been recently called a Pocket Switched Network [HCS$^+$05]. Communication services that rely on this type of data transfer will strongly depend on human mobility characteristics and on how often such transfer opportunities arise. Therefore, they will require fundamentally different networking protocols than those used in the Internet. Since two (or more) ends of the communication might not be connected simultaneously, it is impossible to maintain routes or to access centralised services such as the DNS.

In order to better understand the constraints of opportunistic data transfer, I chose to conduct real-world deployments of devices to members of various communities, allowing me to determine the effects of users' mobility patterns on the prevalence of networking opportunities. I used Intel iMotes to collect connection opportunity data and mobility statistics. I have conducted experiments in various environments including conferences, campuses, small cities like Cambridge and also big metropolitans like Hong Kong. For some experiments, I also used static nodes to mimic infrastructure or city hot spots and to provide approximate location information. A lot of research in MANET or DTN are evaluated on simple mobility models, such as random way point, which are most likely unrealistic. I believe my experimental deployments and data collection are very important for the research community by providing real-life mobility traces for theoretical evaluation. Some of our datasets are now available online in the CrawDad wireless database [KHA04], and are used by many researchers.

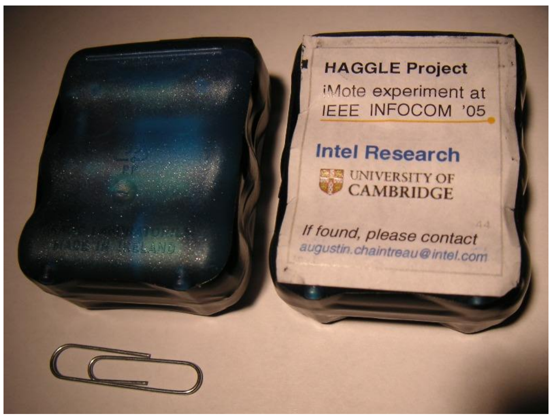
I analyse five datasets from iMote experiments and five external datasets. I define the inter-contact time as the time between two transfer opportunities, for the same devices. I observe in the traces that the inter-contact time distribution follows a heavy-tailed distribution over a range of 10 minutes to 1 day. Inside this range the inter-contact time distribution can be compared to that of a power-law.

$$p(x) = Cx^{-\alpha} \tag{2.1}$$

Distributions of the form (2.1) are said to follow a power law. The constant α is called the exponent or coefficient of the power law. (The constant C is mostly uninteresting; once α is fixed, it is determined by the requirement that the distribution p(x) sum to 1.) To reveal the power-law form of the distribution it is usually better to plot the histogram on logarithmic scales [New05]. I study the impact of those large inter-contact times on the actual performance and theoretical limits of a general class of opportunistic forwarding algorithms that I call "naive" or "oblivious" forwarding algorithms. Algorithms in this class do not use the identities of the devices that are met, nor the recent history of the contacts, nor the time of day, in order to make forwarding decisions. Instead, forwarding decisions are based on forwarding rules statically defined that bound the number of data replicas, or the number of hops.



(a) iMote with battery


HAGGLE Project
iMote experiment at IEEE INFOCOM '05
Intel Research
UNIVERSITY OF CAMBRIDGE
If found, please contact augustin.chaintreau@intel.com

(b) iMote package

Figure 2.1: iMote for the experiment

Based on experimental observations, I develop a simplified model of opportunistic contact between human-carried wireless devices. I do not claim that this model is satisfactory to accurately predict the performance of different forwarding algorithms. It rather serves my purpose, which is to demonstrate how heavy-tail inter-contact times influence the performance of naive forwarding algorithms, and how these should be configured to offer reasonable guarantees.

The rest of the Chapter is structured as follow. I first describe the platform and deployment of the iMote experiments (Section 2.2), then I give a brief summary about the iMote datasets I collected and also some external datasets collected by other research groups that I will use in this dissertation (Section 2.3). After studying the pair inter-contact time distribution (Section 2.4), I present analytical results about forwarding in PSNs with power-law inter-contact time distributions (Section 2.5). As a complement to the theoretical work, I also empirically study the limitations of the "oblivious" forwarding in these real scenarios (Section 2.6). Before the conclusion of this Chapter, I also discuss related work (Section 2.7).

2.2 iMote Experiments

In this section I use an experiment conducted within a group of conference attendees to represent the general features of the iMote experiment. Other experiments follow a similar setup and deployment approach, which involved different participants and environments.

The device used to collect connection opportunity data and mobility statistics in this experiment is the Intel iMote. This is a small platform designed for embedded operation, comprising an ARM processor, Bluetooth radio, and flash RAM, and is shown with a CR2 battery in Figure 2.1(a). I packaged these devices in a dental floss box, as shown in Figure 2.1(b), due to their ideal size, low weight, and hard plastic shell.

Fifty-four of these boxes were distributed to attendees at the IEEE Infocom conference in Miami in March 2005 (which had eight hundred attendees in total). The volunteers were asked to keep the iMote with them for as much of their day as possible. Volunteers were chosen to belong to a wide range of organisations — more than thirty were represented. To assure the participants of their anonymity, I did not record the MAC address of the iMotes that they were given, instead only recording an uncorrelated number printed on the outside of the box, so that I could perform the logistics of distribution and collection. Of the fifty-four iMotes distributed, forty-one yielded useful data, eleven did not contain useful data because of various failures with the battery and packaging, and two were not returned.

The iMotes were configured to perform a Bluetooth baseband layer "inquiry" discovering the MAC addresses of other Bluetooth nodes in range, with the inquiry mode enabled for five seconds. Despite the Bluetooth specification recommending that inquiry last for ten seconds, preliminary experiments showed that five seconds is sufficient to consistently discover all nearby devices, while halving the "battery-hungry" inquiry phase. Between inquiry periods, the iMotes were placed in a sleep mode in which they respond to inquiries but are not otherwise active, for a duration of 120 seconds plus or minus twelve seconds in a uniform random distribution. The randomness was added to the sleep interval in order to avoid a situation where iMotes' timers were in sync, since two iMotes performing inquiry simultaneously cannot see each other. However, I still expect iMotes to fail to see each other during inquiry around four percent of the time (when they are doing inquiry at the same time).

The results of inquiry were written to flash RAM. Since flash capacity is limited (64K for data), it is impossible to store the full results of each inquiry without running the risk of exhausting the memory. Instead, I decided to record "contact periods". This is achieved by maintaining an "in-contact" list comprising the Bluetooth MAC addresses of the nodes that are currently visible. When a device on this list stops responding to inquiries, I store a record of the form $\{$MAC, start time, end time$\}$. Preliminary tests revealed the following problem: Bluetooth devices on a specific brand of mobile phone did not show up consistently during inquiries (and increasing the inquiry period to ten seconds did not help). Therefore, a small number of nodes were causing the memory to fill too quickly. To avoid this problem, I keep a device in the "in-contact list" even if it is not seen for one inquiry interval. If it comes back in-contact on the next interval, nothing is stored. If it does not, a record is stored as normal. This solves the problem, at the expense of not being able to detect actual cases where a node moved out of range during one two-minute period, and back into range for the next two-minute period.

2.3 Experimental Datasets

In this section, I summarise the features of all the experimental datasets which I will use in this chapter and the rest of this dissertation, which include five iMote datasets and five external datasets involve Bluetooth or WiFi. The characteristics of the iMote datasets, explained below, are shown in Table 2.1.

Experimental dataset	Cambridge04	Infocom05	Hong Kong	Cambridge05	Infocom06
Device	iMote	iMote	iMote	iMote	iMote
Network type	Bluetooth	Bluetooth	Bluetooth	Bluetooth	Bluetooth
Duration (days)	3	3	5	11	3
Granularity (seconds)	120	120	120	600	120
Number of Experimental Devices	12	41	37	54	98
Number of internal contacts	4,229	22,459	560	10,873	191,336
Average # Contacts/pair/day	10	4.6	0.084	0.345	6.7
Number of External Devices	148	264	868	11,357	14,036
Number of external contacts	2,441	1,173	2,507	30,714	63,244

Table 2.1: Characteristics of the five iMote datasets

- In *Cambridge04*, the data was obtained from twelve doctoral students and faculty comprising a research group at the University of Cambridge Computer Lab. This is an early experiment and hence has small participant population.
- In *Infocom05*, the devices were distributed to approximately fifty students attending the Infocom student workshop. Participants belong to different social communities (depending on their country of origin, research topic, etc.). However, they all attended the same event for 4 consecutive days and most of them stayed in the same hotel and attended the same sessions (note, though, that Infocom is a multi-track conference).
- In *Hong Kong*, the people carrying the wireless devices were chosen independently in a Hong Kong bar, to avoid any particular social relationship between them. These people were invited to come back to the same bar after a week. They are unlikely to see each other during the experiment.
- In *Cambridge05*, the iMotes were distributed mainly to two groups of students from University of Cambridge Computer Laboratory, specifically undergraduate year1 and year2 students, and also some PhD and Masters students. In addition to this, a number of stationary nodes were deployed in various locations that were expected many people to visit, such as grocery stores, pubs, market places, and shopping centers in and around the city of Cambridge, UK. However, the data from these stationary iMotes will not be used in this chapter. This dataset covers 11 days.
- In *Infocom06*, the scenario was similar to *Infocom05* except that the scale is larger, with 80 participants. Participants were selected so that 34 out of 80 form 4 subgroups by academic affiliations. In addition, 20 long range iMotes were deployed at several places in the conference site to act as access points. However, the data from these fixed nodes is also not used in this chapter.[2]

[2]In this chapter, I do not use Cambridge05 and Infocom06 data because they were unavailable when I was

Table 2.2 summarises the characteristics of the four external experiments (but five datasets): UCSD [MV05], Dartmouth College [HKA04], University of Toronto [SCP$^+$04], and MIT Reality Mining project [EP06]. I name them *UCSD*, *Dartmouth*, *Toronto*, and *MIT* respectively.

User Population	Toronto	UCSD	Dartmouth	MIT BT	MIT GSM
Device	PDA	PDA	Laptop/PDA	Cell Phone	Cell Phone
Network type	Bluetooth	WiFi	WiFi	BT	GSM
Duration (days)	16	77	114	246	246
Granularity (seconds)	120	120	300	300	10
Devices participating	23	273	6648	100	100
Number of internal contacts	2,802	195,364	4,058,284	54,667	572,190
Average # Contacts/pair/day	0.35	0.034	0.00080	0.022	0.23
Recorded external devices	N/A	N/A	N/A	N/A	N/A
Number of external contacts	N/A	N/A	N/A	N/A	N/A

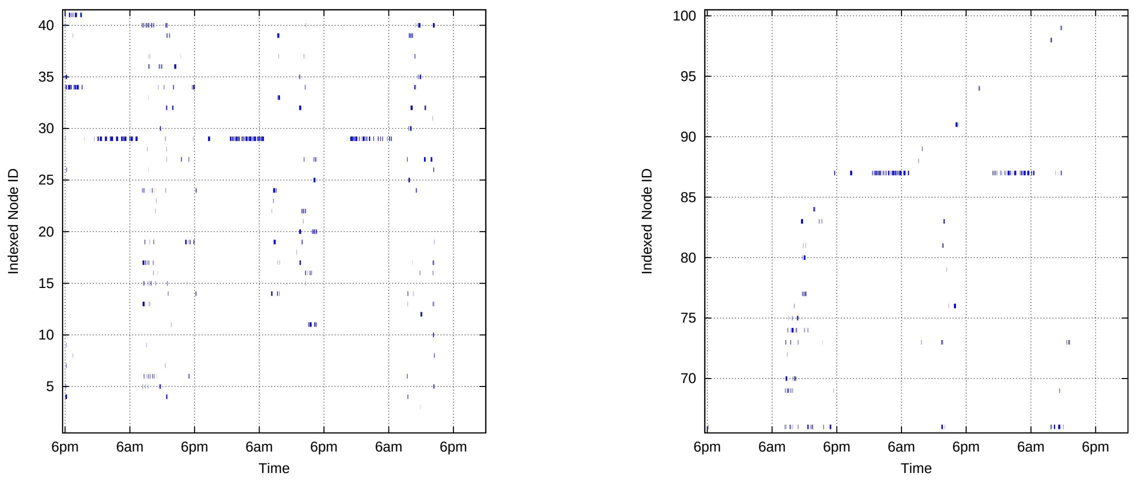
Table 2.2: Comparison of data collected in the external experiments.

UCSD and Dartmouth make use of WiFi networking, with the former including client-based logs of the visibility of access points (APs), while the latter includes SNMP logs from the access points. The durations of the different logs are three and four months respectively. Since we required data about device-to-device transmission opportunities, the raw datasets were unsuitable for our experiment and required pre-processing. For both datasets, I made the assumption that mobile devices seeing the same AP would also be able to communicate directly (in ad-hoc mode), and created a list of transmission opportunities by determining, for each pair of devices, the set of time regions for which they shared at least one AP.

The traces from the Reality Mining project at MIT Media Lab include records of visible Bluetooth devices and GSM cell towers, collected by 100 cellphones distributed to student and faculty on the campus during 9 months. I treat these sets of contacts as two different datasets. For the GSM part, I have assumed, as done above, that two devices are in contact whenever they are connected with the same cell tower.

Unfortunately, this assumption introduces inaccuracies. On one hand, it is overly optimistic since two devices attached to the same (WiFi or GSM) base station may still be out of range of each other. On the other hand, the data might omit connection opportunities, e.g., when two devices pass each other at a place where there is no instrumented access point. Another potential issue with these datasets is that the devices are not necessarily co-located with their owners at all times (i.e. they do not always characterise human mobility). Despite these inaccuracies, these traces are a valuable source of data spanning many months and including thousands of devices. In addition, considering two devices connected to the same base station being potentially in contact is not altogether unreasonable. These devices would indeed be able to communicate

doing the analysis. But in Chapter 3, we will see that users in similar environments (e.g. two Infocom experiments) exhibit similar statistical properties.



(a) iMote with battery (b) iMotes package

Figure 2.2: Data on contacts seen by an iMote: other iMotes (left) and all other device types (right).

locally through the base station without using end-to-end connectivity, or even by using the Internet.

The traces from the University of Toronto have been collected by 20 Bluetooth-enabled PDAs distributed to a group of students. These devices performed a Bluetooth inquiry each 100 seconds and this data was logged. This methodology does not require devices to be in range of any AP in order to collect contacts, but it does require that the PDAs are carried by subjects and that they have sufficient battery life for them to participate in the data collection. Data may be collected over a long period if devices are recharged. The dataset I use comes from an experiment that lasted 16 days.

2.4 Inter-contact Time Analysis

2.4.1 Definitions

I am interested in how the characteristics of transfer opportunities impact data forwarding decisions. In this chapter, I focus on how often such opportunities occur. I decided not to attempt to analyse how much data can be transported for each of them and loss rates [3], as these strongly

[3]Wireless links are lossy, and loss rates are hard to predict, because signal propagation is complex in realistic environments (multi-path fading, reflective obstacles, etc.). In this thesis, I presume that a node may forward packets to another during a recorded “contact” in my measurements. I may underpredict the message count required to delivery the data because of the possible link-layer retransmission to recover from losses. But because I want to focus on the impact of human mobility on forwarding algorithms in this thesis, I temporarily do not consider these physical-layer details and leave them for future work.

depends on factors such as the transmission protocol, the antennas used, and other factors that could be modified to provide improved transmission performance. In my analysis in Section 2.5, I address two extreme cases corresponding to lower and upper bounds for the amount of data that could be transferred in each connection opportunity.

I define the *inter-contact time* as the time elapsed between two successive contacts of the same devices. Inter-contact time characterises the frequency with which packets can be transferred between networked devices; it has rarely been studied in the literature. The behaviour of inter-contact times is important when considering the delay experienced by packets in a PSN. This is the time a node has to wait to get in contact with a specific node (as seen immediately after losing contact with that node). The nature of the distribution will affect the choice of suitable forwarding algorithms to be used to maximize the successfull transmission of messages in a bounded delay. Two remarks must be made at this point:

First, the inter-contact time is computed once at the end of each contact period, as the time interval between the end of this contact and the begin of the next contact with the same devices[4]. Another option would be to compute the remaining inter-contact time seen at any time, i.e at time t, for each pair of devices: the remaining inter-contact time is the time it takes after t, before a given pair of devices meet again (a formal definition is given in Section 2.5). Inter-contact time and remaining inter-contact time have different distributions, which are related, for a renewal process, via a classical result known as the waiting time paradox (see p.147 in [Bre99]). A similar relation holds for stationary processes, in the theory of Palm Calculus (see p.15 in [BB03]). I choose to study the first definition of "inter-contact time seen at the end of a contact period", as the second gives too much weight to large values of inter-contact times. In other words the definition that was chosen is the most conservative one in the presence of large values.

Second, the inter-contact time distribution is influenced by the duration and the granularity of the experiment. Inter-contact times that last more than the duration of the experiment cannot be observed, and inter-contact times close to the duration are less likely to be observed. In a similar way, inter-contact times that last less than the granularity of the measurement (which ranges from two to five minutes among different experiments) cannot be observed.

Another measure of the frequency of transfer opportunities that could be considered, is the inter-any-contact, i.e. for a given device, the time elapsed between two successive contacts with any other device. This measure is very much dependent on the deployment of wireless devices and their density during the experiment, as it characterises time that devices spend without meeting any other device.

[4]Inter-contact starting after the last contacts recorded for this pair of devices were not included.

2.4.2 Inter-contact Time Characterisation

I study the empirical distribution of the inter-contact times obtained for all experiments shown in Figures 2.3, 2.4, and 2.5.

For all plots, an empirical distribution of the inter-contact times was first computed separately for each pair of devices that met at least twice. It is hard to study the characteristics of the distributions for all pairs individually, because there are many such distributions, and some of them may only include a few observed values. This is why I follow a two-step approach: First, I present the distribution obtained when all pairs distributions are combined, each with an equal weight, in a distribution that I call the aggregated distribution. Second, I use a parametric model motivated by this first part and estimate the parameters of the individual distribution for each pair.

1) *Aggregated distribution*: Figures 2.3, 2.4, and 2.5 present the aggregated distribution for different datasets. All plots show the complementary cumulative distribution function, using a loglog scale.

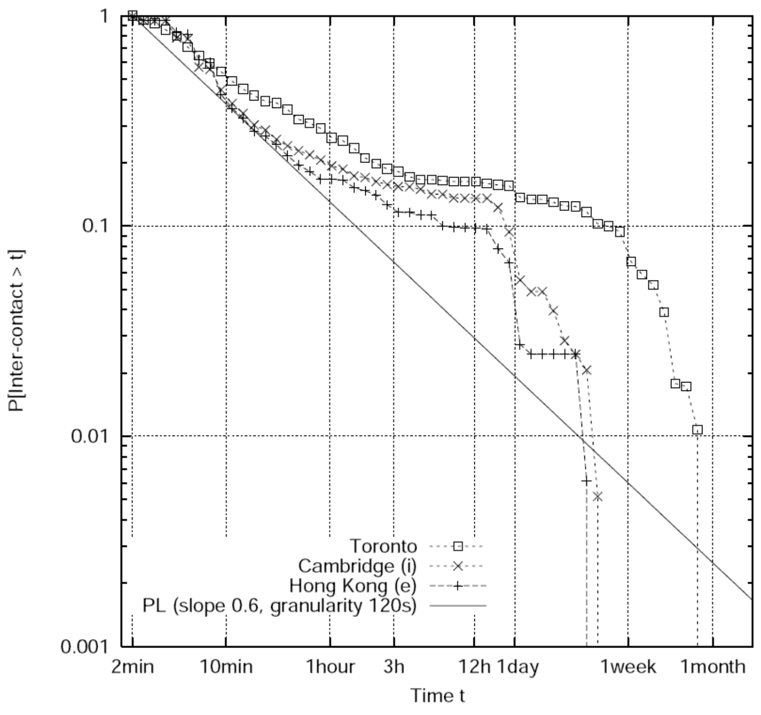


Figure 2.3: Aggregated distribution of the inter-contact time in *Cambridge*, *Hong Kong* and *Toronto* experiments

For iMote experiments, “(i)” indicates that the dataset shown is obtained using internal contacts only, while “(e)” indicates that the dataset shown includes only external contacts. For the first two iMote experiments (labeled *Cambridge* and *Hong Kong*) I present only one case here (corresponding respectively to internal and external contacts). They are shown in Figure 2.3, which also includes the distribution obtained among pairs of experimental devices in the trace from the University of Toronto. Distributions belonging to the iMote-based experiment at *Infocom05* are shown in Figure 2.4 , where distributions associated with internal and external contacts have been plotted separately for comparison. Figure 2.5 presents the distribution of inter-contact computed using traces from experiments other than ours.

Let us first note that, although inter-contacts are short in most cases, the occurrence of large inter-contacts is far from negligible: in the three iMote based experiments (*Cambridge*, *Infocom05* and *HongKong*), 17 to 30% of inter-contact times are greater than one hour, and 3 to 7% of all inter-contacts are greater than one day. In the Toronto datasets, 14% of inter-contacts last more than a day, and 8% more than a week. These large inter-contacts are even more present in the traces collected in *UCSD*, *Dartmouth* and *MIT*, the most extreme case being the *MIT* trace using Bluetooth sightings, where up to 60% of the inter-contacts observed are above one day. The variation between datasets is significant. It can be expected given the diversity of communication technologies and populations studied, as well as the impact of experimental conditions (granularity, duration). But they also present common properties that I now discuss in more detail.

I now concentrate on the region between 10 minutes and one day. In this region, all datasets exhibit the same characteristics: the cumulative distribution function (CCDF) is slowly varying, it is lower bounded by the CCDF of a power-law distribution, that may in some cases be a good approximation. This contradicts the exponential decay of the tail which characterises the most common mobility models found in the literature, and I prove in the next section that it can have a significant impact on the performance of opportunistic networking algorithms.

To justify the above claim, I studied the quantile-quantile plot comparison between the empirical distribution found and three parametric models (exponential, log-normal, and power-law). An example is shown in Figure 2.6 for the distribution based on internal contacts during the *Infocom05* experiment. All parametric models have been set to take the same median value as the empirical distribution. I also normalise the power-law to fit the granularity t=120 seconds, and the log-normal distribution such that the logarithm of both the empirical variable and the model have the same variance. Not surprisingly, we observe that the three models deviate significantly from the empirical findings for values above one day. As expected, the exponential distribution is far from the empirical ones, the quantile for the log-normal distribution deviates from the empirical case by a non negligible factor. The power-law distribution, by opposition, remains close to the empirical one for values up to 18 hours, and it seems to be the most appropriate model to apply. In other datasets, the power-law may sometimes not match the empirical findings as well as in this example, but among these three models it is always the closest to the empirical distribution. For values above one day, I expect models with additional parameters (e.g. following a Weibull distribution) to improve the match with the empirical distribution, but that is beyond the scope of this chapter.

The most notable difference I observe between datasets is that the fit with a power-law is better

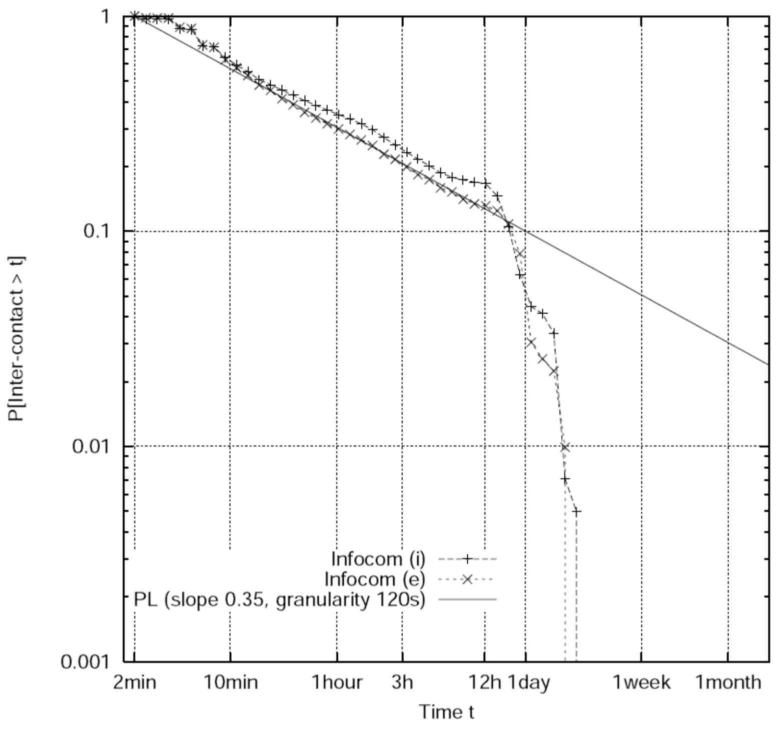


Figure 2.4: Aggregated distribution of the inter-contact time in Infocom05 experiment

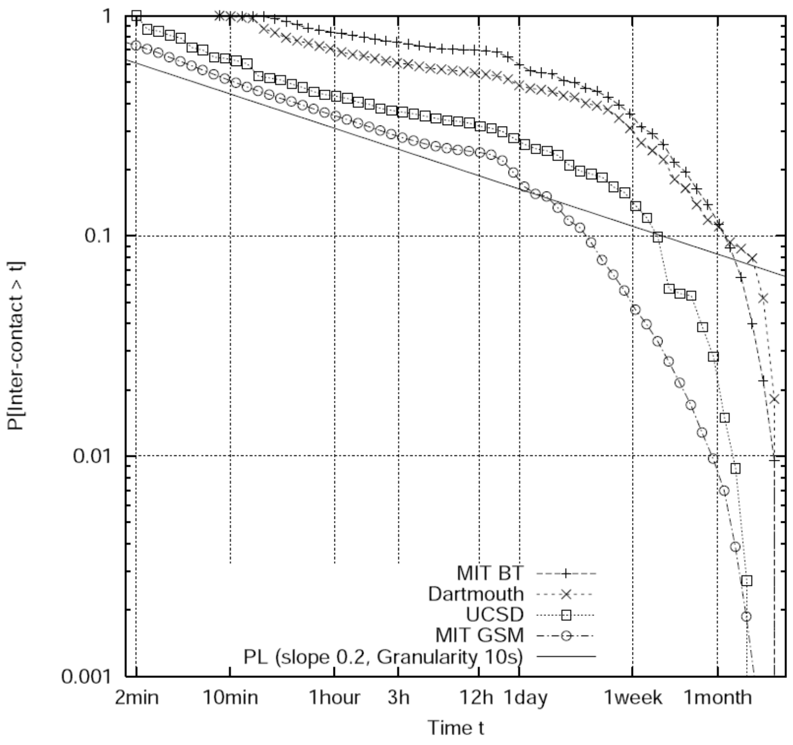


Figure 2.5: Aggregated distribution of the inter-contact time in UCSD, Dartmouth and *MIT* experiments

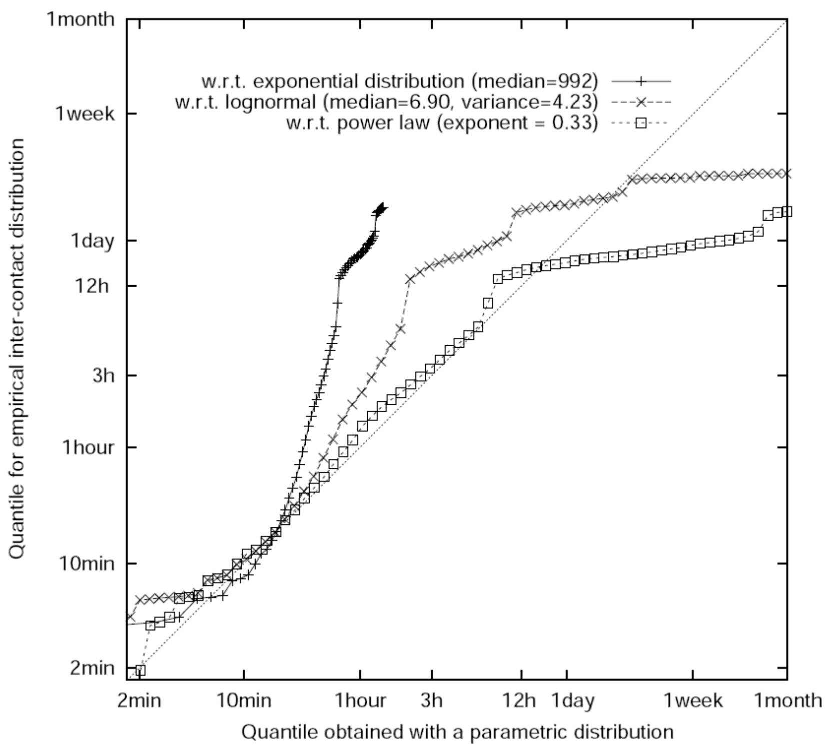
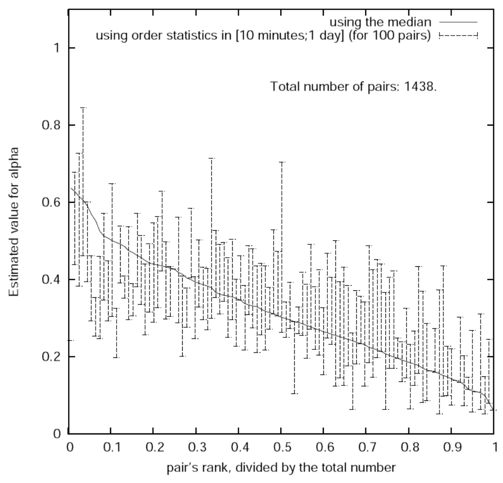


Figure 2.6: Inforcom05: Quantile-quantile plot of comparison between the aggregated distribution of the inter-contact time and three parametric models

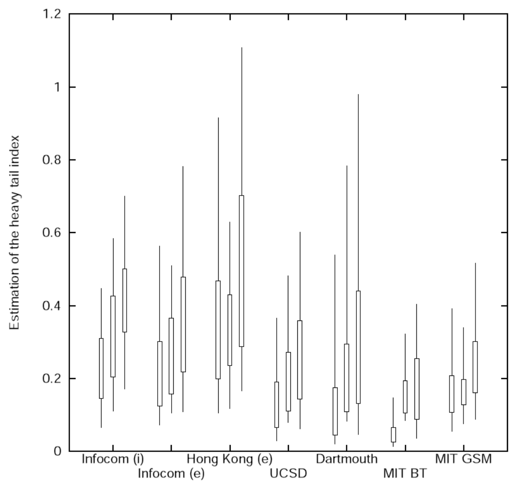
for the datasets that contain the largest number of points, such as in Figure 2.4 and Figure 2.5. I also observe that the coefficient of the power-law that is a lower bound on the range [10 minutes; 1 day] is different between datasets: this is 0.6 for the iMote experiments at *Cambridge* and *Hong Kong*, as well as for Toronto datasets, 0.35 for the iMote based experiment at *Infocom05*, and 0.2 for traces collected in UCSD, Dartmouth and MIT. In all cases, it is below 1. The value of this coefficient, which is also called the heavy-tail index, is critical for the performance of opportunistic forwarding algorithms, and I discuss it further below.

Figure 2.4 shows that the distribution is almost unchanged if one considers internal or external contacts. The same observation was made for other iMote experiments, except for the experiment conducted in Hong Kong where very few internal contacts were logged. Some variations of the heavy-tail index have been observed depending on the time of the day.

2) *Individual distribution for each pair*: So far I have studied the aggregated distribution where all pairs have been combined together, and I found that it can be approximated by a power-law for values up to 1 day. In this section, I assume that this claim can be made individually for all pairs, although the parameter of this power-law, also called the heavy-tail index, may be different among them. This approach allows us to study the heterogeneity between pairs via a single parameter, some of these results also measure the accuracy of the above assumption for each pair.



(a) Infocom05



(b) All Experiments

Figure 2.7: Estimation of the heavy-tail index of the power-law applied separately for each pair for *Infocom05* (left) and summary of results obtained in all datasets (right).

Estimator for the heavy-tail index Let us consider a pair of nodes. The sample of the inter-contacts observed for this pair will be denoted by $X_1, \ldots, X_n$, its order statistics by $X_{(1)} \leq \ldots \leq X_{(n)}$, and its median value by m. All times will be given in seconds. If we assume that this sample follows a power-law with granularity 120s and heavy-tail index α, we have: $P[X \geq x] = (x/120)^{-\alpha}$, such that an estimator of α based on the samples' median m is given by:

$$\breve{\alpha} = \frac{ln(2)}{ln(m) - ln(120)} \tag{2.2}$$

More generally one can consider all order statistics X(i) that fit in the range [10 minutes; 1 day] and estimate α based on each of them. It creates a collection of estimators for the value of α, as follows:

$$\{\frac{ln(n) - ln(n-i)}{ln(X_{(i)}) - ln(120)} \mid 600 \leq X_{(i)} \leq 86400, i < n\} \tag{2.3}$$

I denote by α_{low} and α_{up} respectively the minimum and maximum values in this set above. It is equivalent to plot the empirical CCDF for this sample in a log-log scale, and bound this CCDF from above and below by two straight lines that go through probability 1 at time value 120s. These slopes would be equal respectively to $-\alpha_{low}$ and $-\alpha_{up}$. By opposition to $\breve{\alpha}$, these two estimators are not centred around the value of α, and they do not converge to this value when the sample becomes large. They rather serve the purpose of a heuristic analysis; they characterise some bounds that are verified by each pair. Note also that, intuitively, the difference $\alpha_{up} - \alpha_{low}$ indicates how the conditional distribution of the sample in this range differs from a pure power-law.

In Figure 2.7(a), I plot the values of $\check{\alpha}$ and the interval $[\alpha_{low};\alpha_{up}]$ for all pairs of iMotes during the experiment conducted at *Infocom05*. One can expect that the coefficient takes different values among pairs, as some participants are more likely to meet often than others. I initially ranked all pairs according to their value for $\check{\alpha}$, in decreasing order. Although I have computed these values for all pairs, I only draw the interval $[\alpha_{low};\alpha_{up}]$ for 100 pairs chose,n arbitrarily according to their rank (one every 14), in order to keep the figure readable. As shown in Figure 2.7(a), estimations of α for different pairs may indeed vary between 0.05 and 1. Between these two extreme values, which are rarely observed, estimates for almost all pairs lie between 0.1 and 0.7 depending on the estimator. Note that all estimates of α are smaller than 1; the only exceptions are the upper estimate α_{up} for three pairs (i.e. less than 0.2% of pairs in this case). The median-based estimate lies in [0.2 ; 0.4] for half of the pairs, the lower estimates lies in [0.14 ; 0.32], and the upper estimate lies in [0.32 ; 0.5] again for half of the pairs.

These results have three major implications: First, the heterogeneity among pairs implies different possible values for α, which are centered around the value already observed when studying the aggregate distribution (i.e. 0.33). Second, the difference between the median estimator and the heuristic bounds I defined above remains within 0.25 except in a few cases. Last, the upper estimate α_{up} almost never goes above 1, which establishes that the inter-contact distribution for each pair is lower bounded in this range by a power-law with a coefficient smaller than 1.

The same results have been obtained for other datasets, and they are summarised in Figure 2.7(b). For each dataset indicated in the bottom, I show the distribution of values obtained among pairs for the three estimators defined above. Each estimator stands for one box-plot: it is, from left to right, $\alpha_{low},\check{\alpha},\alpha_{up}$; the thick part indicates the values found in 50% of the pairs, the thin part contains the region where 90% of the pairs are found.

In the *Hong Kong* and *Dartmouth* datasets, where contacts are sparser, inter-contact samples for each pair contain fewer values. As a consequence, the difference between estimators can grow significantly. I even observe that α_{up} goes slightly beyond 1 for 10% of the pairs in *Hong Kong* dataset, although it is probably an artefact of my conservative estimate.

Correlation: We study the auto-correlation coefficients to see how the value of the inter-contact time may depend on the previous values for the same pair. The results are shown in 2.8 for all order k up to 50. Since the inter-contact time distribution usually has no finite variance, we computed the correlation coefficient on the values of the logarithm of the inter-contact times. Note that a correlation coefficient was computed for each pair, we present for all order k the average value we observed among all pairs, as well as the interval containing 50% and 90% of the centred values (respectively, in the thick box and the thin bar).

In the *Infocom05* dataset, the variation of the coefficient among pairs is quite important, although most pairs remain reasonably non-correlated (the thick box remains always less than 0.30 away from zero). Overall we observe a slightly negative correlation on average over all pairs, which reduces as k grows. Correlation coefficients are smaller when the dataset is large (as seen for example in the *MIT* GSM trace shown here, as well as for all other long traces).

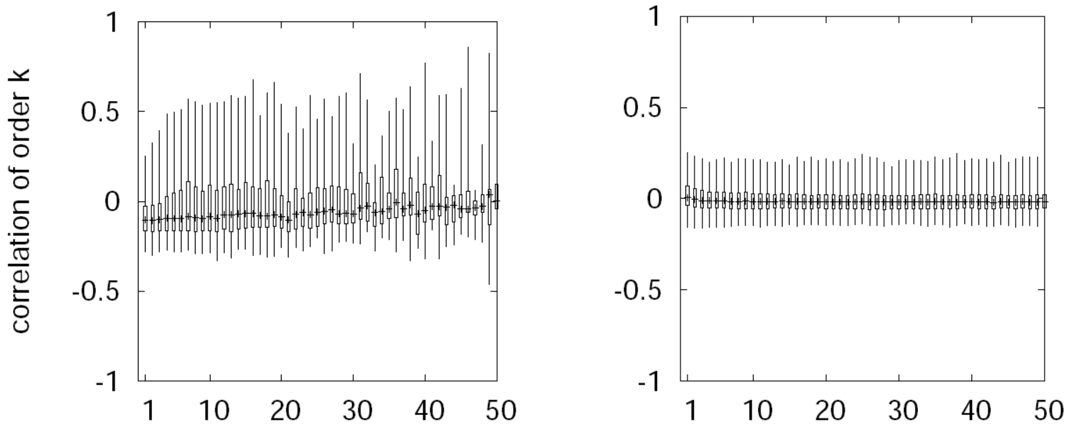


Figure 2.8: Correlation coefficients for the sequence of inter-contact times: for all pairs of iMotes in the *Infocom05* dataset (left), and for all pairs of devices in *MIT* GSM dataset (right)

This tends to indicate that these coefficients for all pairs would be closer to zero if the iMote experiment could be done with a longer duration, and that the sample of inter-contacts collected for each pair was bigger.

Based on the above results, I assume in the next section that the inter-contact time distribution follows a power-law for each pair. To simplify the analysis I assume in addition that the coefficient is the same for all pairs, that the sequence of inter-contact times is i.i.d. (i.e. correlation coefficient are null) and that they are independent between pairs. This simplification allows me to characterise the performance of forwarding algorithms quite generally. Some of the results I present can be extended to stationary ergodic sequences, but that is left for future work.

2.5 Forwarding with Power Law-based Opportunities

In this section, I present a summary about forwarding with power law-based opportunities. But as mathematical analysis is not a main contribution of this thesis, I recommand readers to refer to our paper [CHC$^+$07] for the details of the proof.

I am interested in a general class of forwarding algorithms, which all rely on other devices to act as relays, carrying data between a source device and a destination device that might not be contemporaneously connected. These relay devices are chosen purely based on contact opportunism and not using any stored information that describes the current state of the network. The only information used in forwarding is the identity of the destination so that a device knows when it meets the destination of a bundle. I call such algorithms “oblivious”, although they could be in reality quite complex and, as we will see, very efficient in some cases.

The following two algorithms provide bounds for the class of algorithm described above:

- wait-and-forward: The source waits until its next direct contact with the destination to communicate.
- flooding: a device forwards all its received data to any device which it encounters, keeping copies for itself.

The first algorithm uses minimal resources but can incur long delays and does not take full advantage of the ad-hoc network capacity. The second algorithm, that was initially proposed in [VB00], delivers data with the minimum possible latency, but does not scale well in terms of bandwidth, storage, and battery usage. In between these two extreme algorithms, there is a whole class of algorithms that play on the number of relay devices to maximise the chance of reaching the destination in a bounded delay while avoiding flooding. The most important reason not to flood is to minimise memory requirements and related power consumption in relay devices, and to delete the backlog of previously sent messages that are still waiting to be delivered, and could be outdated. Some strategies, based on time-outs, buffer management, limit on the number of hops and/or duplicate copies have been proposed (see [VB00, CM01, DFL01]) to minimise replication and backlog.

I do not include the contact time representing the duration of each contact in this model , assuming that each contact starts and ends during the same time slot. This is justified here by the fact that I am interested in a model that accounts for consequences of large values of the inter-contact time. It was observed (see [HCS$^+$05]) that the contact time distribution is also heavy-tailed, but it takes smaller values, by several orders of magnitude, than the ones of the inter-contact time. Even if we do not explicitly model the contact time (each contact lasts one time slot), we need to take into consideration the fact that a contact may last long enough to transmit a significant amount of data. I then introduce two situations:

- the *short contact case* : where only a single data unit of a given size can be sent between two devices during each contact.
- the *long contact case* : where two devices in contact can exchange an arbitrary amount of data during a single time slot.

These two cases represent a lower and an upper bound for the evaluation of bandwidth. The number of data units transmitted in a contact (whether short or long) is defined as a data bundle. The long and the short case differ from a queuing standpoint. In the long contact case, the queue is emptied any time a destination is met. In the short contact case, only one data unit is sent and therefore, data can accumulate in the memory of the relay device.

At this stage, I have established the following results for the class of so-called “oblivious” forwarding algorithms defined in the long contact case :

- For $\alpha > 2$ any algorithm from the class I considered achieves a delay with finite mean.

- If $1 < \alpha < 2$, the two-hop relaying algorithm, introduced by [GT02], is not stable in the sense that the delay incurred has an infinite expectation. It is however still possible to build a naive forwarding algorithm that achieves a delay with finite mean. This requires that a number of m duplicate copies of the data are produced and forwarded, where m must be greater than $\frac{1}{\alpha-1}$, and the network must contain at least $\frac{2}{\alpha-1}$ devices.
- If $\alpha < 1$, none of these algorithms, including flooding, can achieve a transmission delay with a finite expectation.

In other words, I have characterised the performance of all these algorithms in the face of extreme conditions (i.e. heavy-tailed inter-contact times). The last case where $\alpha < 1$ corresponds to the most extreme situation, and the result I provide in this case seems at first unsatisfactory: none of the algorithms I have introduced can guarantee a finite expected delay. To make the matter worse, this case where $\alpha < 1$ seems to be typical of the inter-contact distribution in the [10 minutes; 1 day] range for all the scenarios I have previously studied empirically. This overall implies that the expected delay for all the scenarios I have discussed before should be at least of the order of one day. Note that this was shown for any forwarding algorithms used, and even when queuing delays in relay devices are neglected.

2.6 Empirical Evaluation of Controlled Flooding Algorithms

I have shown above that using devices met opportunistically to relay a message toward its destination improves significantly the chance of delivering this message, and reduces its delivery delay. Indeed I established a stronger result: in all datasets, a small number of intermediate hops are enough to reach most of the optimal paths (minimum latency). This indicates that designing algorithms to forward messages based on a simple Time-To-Live (TTL) can sometimes be very successful.

I use the forwarding algorithm emulator which will be described more detail in Chapter 4 to analyse practical controlled flooding algorithms. The controlled flooding algorithms work as follows: each message received is sent several times, to the first devices met, until one of the TTL counters (limiting the number of hops, the number of copies sent locally, or the time) reaches zero. I study their cost, and their success delivery.

There are several reasons to believe that focusing on such simple forwarding algorithms is a natural first step to follow. This problem is easy to formulate and leads to a good characterisation based on the real-life traces I presented. It can serve as a baseline to propose and evaluate more complex forwarding techniques. In addition, such TTL-based techniques may be needed to eliminate old data from the system, even in the presence of smarter forwarding algorithms. Note also that it does not necessarily mean that I am restricted to inefficient techniques. Indeed, models of human interaction networks were recently proposed in which even simple algorithms are proved to benefit from unknown but regular human behavior (See [Kle06] and references therein).

2.6.1 Controlled Flooding

Controlled flooding algorithms or as I also call them Multiple-Copy-Multiple-Hop (MCP) [5] algorithms work as follow. When created, a data packet is given a time-TTL value, as well as a hop-TTL integer value. At each intermediate node, the hop-TTL value is decreased by 1 when the packet is received. A packet with hop-TTL=1 can only be delivered to the destination directly and can not be forwarded to intermediate nodes. A packet with hop-TTL value above 1 is replicated *m* times: one copy is kept by this node, and *m*-1 are sent to the first devices met that did not receive this packet before. The time-TTL [6] value acts as a time-out; nodes decrease the counter with time and discard the packet when this counter becomes zero.

Note that if $m = 1$ the packet will not leave the source unless the destination is met directly, as well as if the hop-TTL is initially set to 1. The initial value given to hop-TTL is also the maximal number of hops that are used by a forwarding path. Note that the two-hop relaying algorithm proposed in [13] would here correspond to hop-TTL=2, *m*=2, time-TTL=∞.

The forwarding algorithm emulator generates 1000 messages over the duration of the experiment, at a time chosen with uniform distribution. The source and the destination are chosen uniformly among all experimental devices. The goal of this emulation is certainly not intended to carry out a full-fledged representative simulation of a real PSN. Instead I found this method suitable for my purposes, which is to compare generally the performance of controlled flooding algorithms to optimal found with the previous methodology.

Note that I have performed emulation on all three datasets and observations are consistent with those shown below.

2.6.2 Performance and Cost

In this section, I evaluate controlled flooding algorithms based on two metrics: (i) the success delivery ratio, which is the probability for a packet created at a source node to reach its destination within a delay smaller or equal than time-TTL ; and (ii) the average cost per message, which is the total number of copies created in the network, divided by the number of messages generated in the sources (i.e. 1000 in this experiment). The statistics about each experiment, including the number of nodes, can be found in Section 2.3.

[5]A more correct version should be MCMH, but because MCP has been used in my previous publications so I just keep it here for consistency.

[6]I consider time TTL here because in some situations the applications can specific the time that a message should stay in the system. For example, a message about a meeting at 10 am is meaningless if it arrives at 6pm of the same day. It also serves as an indicator of how sparse a network is.

The performance of all controlled flooding algorithms lies between that of *wait-and-forward* (i.e. the case hop-TTL= 1 or m = 1) and *flooding* (i.e. no TTL, messages are sent to every encounter). The success delivery ratio for these algorithms was established for any time-TTL in the previous section.

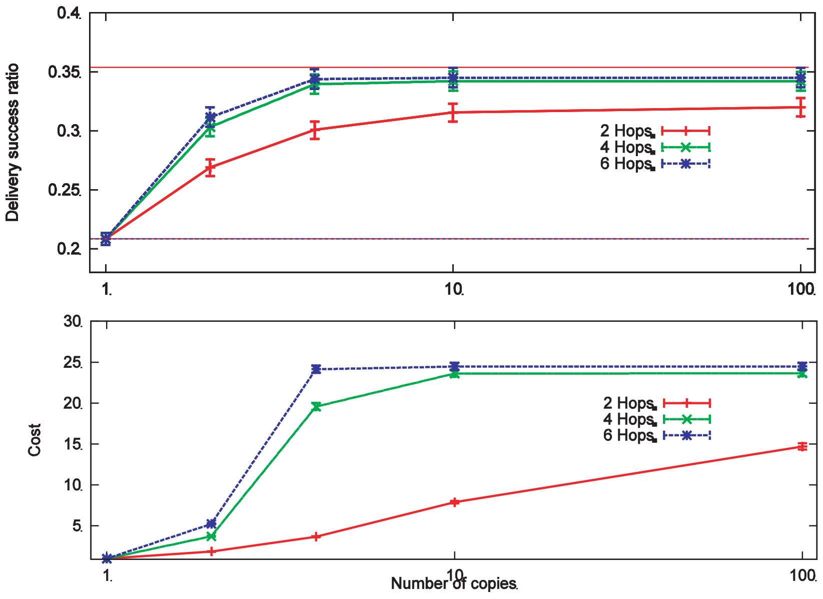


Figure 2.9: Success delivery (top) and Average Cost (bottom) observed with *Infocom05* for several controlled flooding schemes: time-TTL was set to 3 hours.

Figure 2.9 shows the performance of controlled flooding on the *Infocom05* dataset for a time-TTL set to 3 hours. The x-axis corresponds to the number of copies created locally at each hop (i.e. *m*). Each plot corresponds to a different value of the hop-TTL. The delivery success ratio is shown at the top along with the performance evaluated with the earlier method for *wait-and-forward*, and *flooding*. [7] In the bottom plot, I show the cost of each algorithm. Both performance and cost are shown with confidence interval describing the minimal and maximal values seen in the 100 runs of the experiment. I observe that the delivery success ratio might vary for different simulations (the difference might grow up to 7%).

The first observation is that both delivery success and cost converge quickly when *m* increases, especially when hop-TTL is greater than 2. For both 4 and 6 hops, I observe only marginal improvement of the delivery success ratio for *m* above 4 (the same observation holds for 2 hops when *m* is increased above 10). Note that the cost of the algorithm still increases a little when *m* is larger than 4, but it is already reasonably close to the asymptotic cost corresponding to flooding. This essentially shows that no controlled flooding algorithm can match the delivery success ratio of flooding without incurring a similar cost. This trade-off leaves room for improvement: an algorithm which has more restriction than controlled flooding, based on an intelligent design,

[7] *wait-and-forward* (*flooding*) is supposed to be at only one point on the graph, but I draw it as a horizontal line to emphasise the lower bound (upper bound).

could succeed reducing this cost with a small impact on the overall delivery ratio.

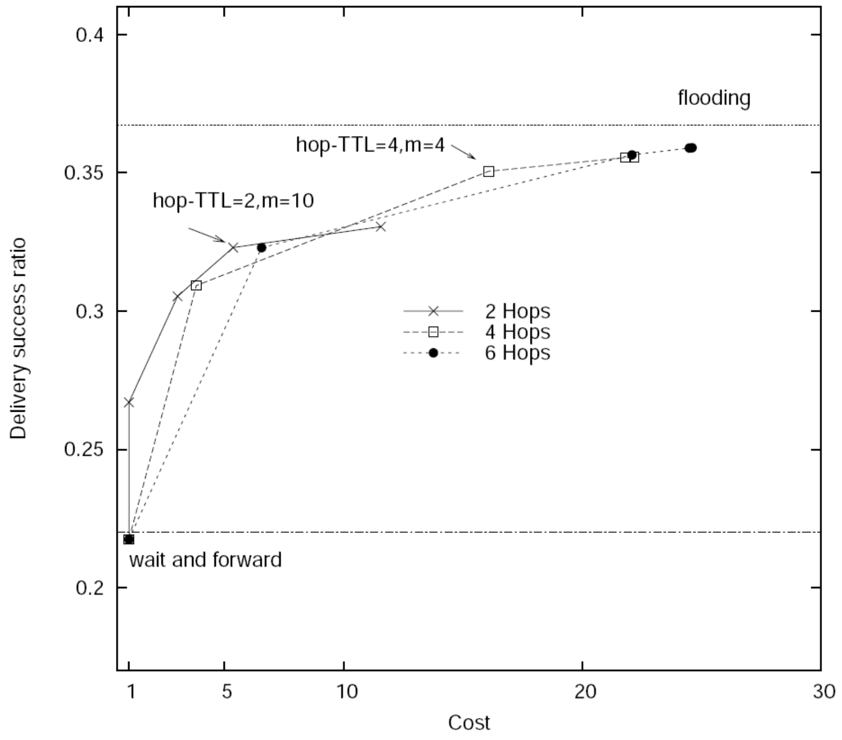


Figure 2.10: Performance-Cost trade-off for time-TTL=3 hours during Infocom05: success delivery obtained as a function of the total number of copies created per message, for different controlled flooding algorithm.

The optimal trade-off area between cost and success delivery ratio is found for values of *m* between 2 and 10, and values of hop-TTL between 2 and 6. Note that the best improvement with regard to the cost is found for the most conservative algorithm: 2 hops and *m*=2 yields an average increase of 5% for the delivery ratio, for a marginal cost. Another 5% average increase may be achieved within a reasonable cost by remaining conservative: either by allowing 6 hops and keeping $m = 2$, or by keeping 2 hops but setting *m* equal to 10. I present this trade-off between average performance and cost in Figure 2.10: the same data points are shown where x-axis corresponds to the cost value, whereas y-axis represents delivery success. The delivery success ratio obtained by wait-and-forward and flooding was shown again to delimit the bound of the area. The value corresponding to *m* following values 1,2,4,10,100 can be read from the left to right on each plot corresponding to each hop-TTL (two data points are explicitly detailed for future reference). The best delivery success ratio is achieved by restricting paths to 2 hops and increases *m* when cost needs to remain moderate (under 6 copies created in total per message). Keeping 2 hops as a maximum and increasing *m* proved to be suboptimal at some point, as algorithms with smaller *m* but longer paths perform better for a similar cost. In the next section, I analyse in more detail the two parameter settings corresponding to these two cases: (hop-TTL=2,*m*=10) and (hop-TTL=4,*m*=4).

Results obtained with other traces are comparable. Because the network is much sparser in the other datasets, I found that setting time-TTL to 3 hours produces a delivery ratio that is too small to be effectively studied. I present the case of Reality Mining, where time-TTL is set to 24 hours, in Figure 2.11. Observations are similar to *Infocom05*: both average performance and cost quickly converge with m, the trade-off between them follows a very similar diagram than the one presented above for *Infocom05*, where 2 hops again yields the best immediate benefit.

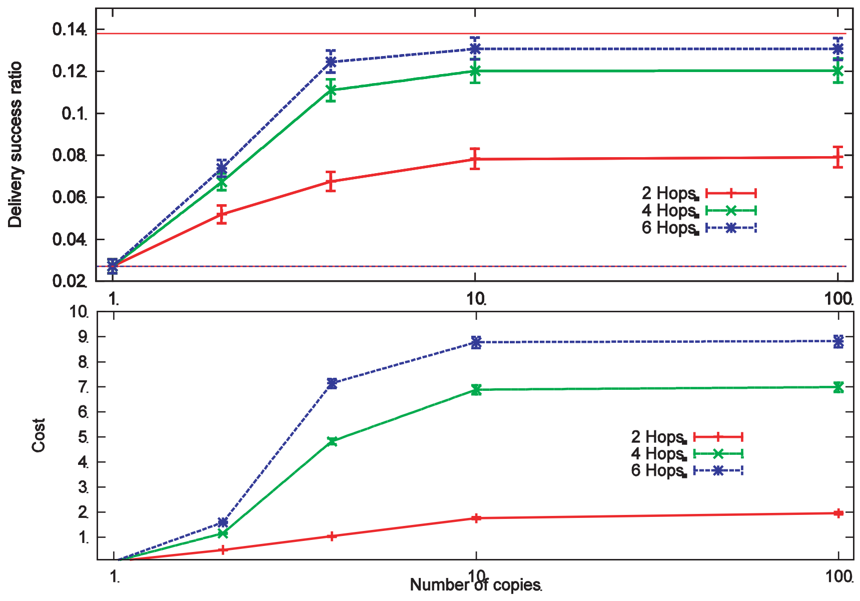


Figure 2.11: Success delivery (top) and total number of copies sent (bottom) in *Reality*, for several controlled flooding schemes: time-TTL is set to 24 hours.

2.6.3 Delay and Impact of Time-TTL

Impact of the hop-TTL and *m* were analysed in the previous section, for a fixed value of the time-TTL. In this section, I focus on two controlled flooding algorithms, (hop-TTL=2, *m*=10) and (hop-TTL=4, *m*=4) that offer different performance-cost trade-off, as shown above. I compare their dependence on the values of time-TTL.

The delivery success with a fixed time-TTL value gives the proportion of packets, which reach their destination within a certain time, hence it also describes the delay distribution of packets forwarded by this algorithm. I plot this distribution in Figure 2.12 for these two different parameter setting.[8] The cost of each algorithm with these values of time-TTL is shown in Figure 2.13. Note that the delivery success for both algorithms follows the same trend at every time scale. The algorithm with 4 hops and smaller value of m remains close to the optimal at all time-scales. The algorithm with 2 hops and large m yields a significant benefit when compared

[8]Because I have plotted the 95th percentile in previous section, here I show the maximum and minimum in the graph for additional information.

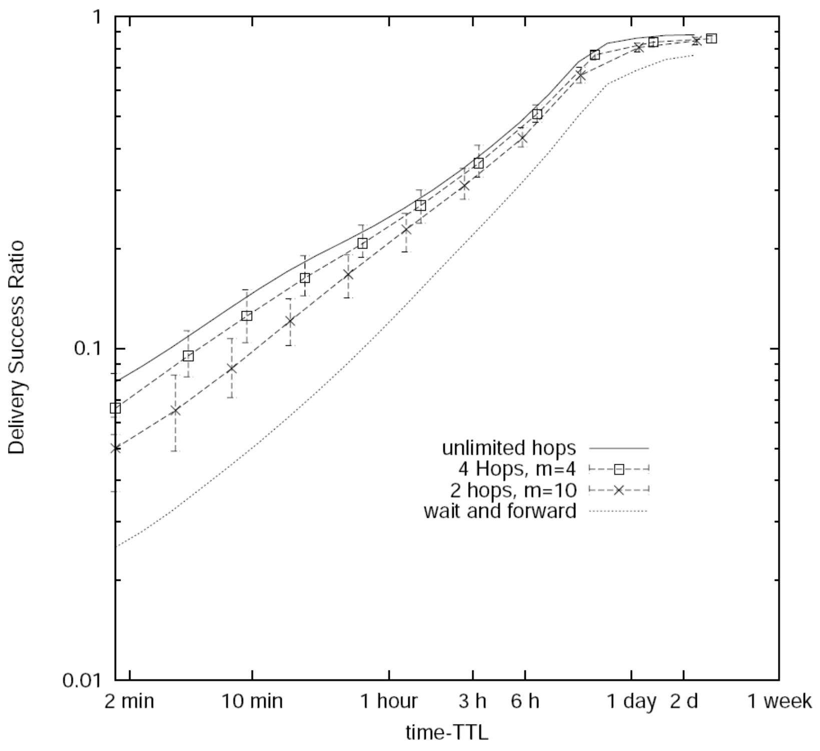


Figure 2.12: Success delivery seen as a function of time-TTL during *Infocom05* for three different controlled flooding parameters.

to wait-and-forward. One thing to notice is that their difference tends to be much reduced when time-TTL is large. Analyzing the cost of both algorithms as a function of the time-TTL confirms that using 2 hops with large m is the best candidate when time-TTL is large, as it allows a similar performance improvement while keeping the number of packets created reasonablely small (under 10).

2.7 Related Work

My opportunistic communication model is related to both Delay-Tolerant Networking and Mobile Ad-Hoc Networking[9]. Research work on MANET, DTN, and more recently PSN [SHCD06] confirms the importance of the problem I address, as several propositions were made to use mobile devices as relays for data transport. Such an approach was considered to enable communication where no contemporaneous path may be found [VB00], to gather information efficiently in a network of low-power sensors [JSB$^+$04, JOW$^+$02b], or to improve the spatial reuse of dense MANETs [GT02, SMS06]. All these work has proved that the mobility model used has a strong impact on the performance of the algorithms they propose.

I did not find any previous work studying the characteristics of inter-contact time for users of

[9]www.dtnrg.org and www.ietf.org/html.charters/manet-charter.html

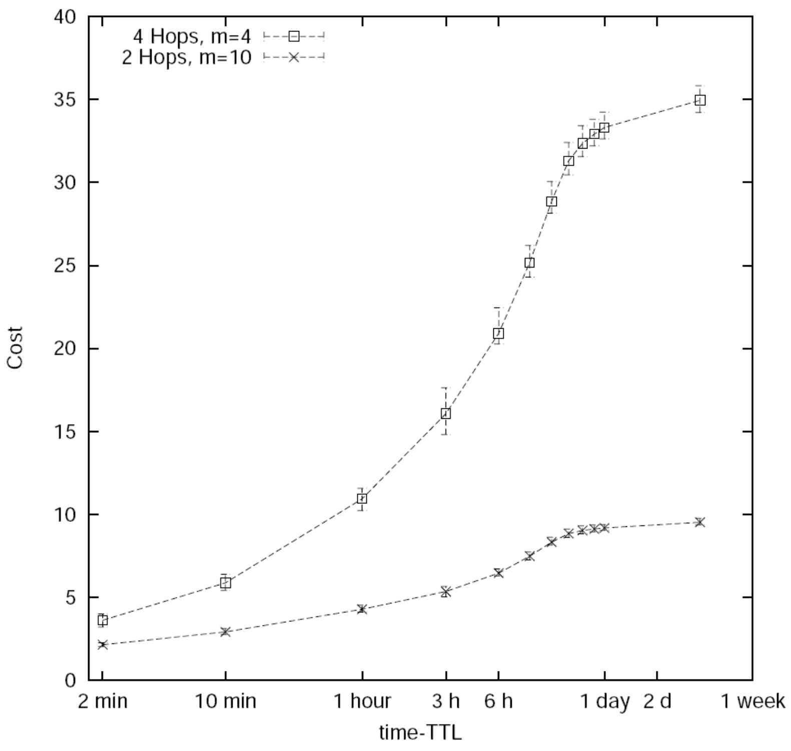


Figure 2.13: Average cost of a message seen as a function of time-TTL during *Infocom05*, for three different controlled flooding parameters.

portable wireless devices. However, I have identified related work in the area of modeling and forwarding algorithms.

A common property of many mobility models found in the literature is that the tail of the inter-contact distribution decays exponentially. In other words, for these models, the inter-contact time is light-tailed. This is the case for i.i.d. locations of devices in a bounded region (as assumed in [GT02]), or in the case of the popular random way point model as demonstrated in [SMS06]. It was shown recently that, by opposition, devices moving according to Brownian motion in a bounded region, exhibit heavy-tailed inter-contact time, with a finite variance (corresponding in my analysis to the case $\alpha > 2$) (see [SMS06] and references therein).

The most relevant work is the algorithm proposed by Grossglauser and Tse in [GT02], further analysed in [SMS06]. The two-hop relay forwarding algorithm was initially introduced to study how the mobility of devices impacts the capacity of the network. My work starts from very different assumptions. Most notably, I do not model the bandwidth limitation due to interference, as I focus only on the delay induced by mobility. However, some of the results that I show could be used to characterise the delay obtained in such contexts.

2.8 Conclusion

I have analysed several network scenarios for opportunistic data transfer among mobile devices carried by humans, using eight experimental datasets. For all datasets, I observe that the inter-contact time between two devices can be approximated by a power-law on the [10 minutes; 1 day] range. I prove in a simple model the following major results: power-law condition may be addressed by naive forwarding algorithms as long as the heavy-tail index of the power-law is greater than 1. When, by opposition, the heavy-tail index is smaller than 1, the expected delay cannot be bounded for any forwarding algorithm of that type, even when one neglects the queuing occurring in each relay device. I have measured a heavy-tail index smaller than 1 in all datasets. As a consequence, the expected delay is at least in the order of one day. [10]

These observations bring new practical recommendations to evaluate the performance of forwarding algorithms. Most of the mobility models commonly used today are characterised by a light tailed inter-contact distribution for any pair of nodes. That seems at odds with the empirical evidence of inter-contact distribution, for values up to 1 day, which is well approximated by a heavy-tail distribution. Some of these models can in theory be modified to account for this last property, this may be a future research direction. Another complementary direction, which is chosen in this chapter, is to directly model opportunities between devices instead of geographical locations. This approach has the advantage that it can be directly compared with a growing set of real-life connectivity traces, now publicly available. I believe that this is a practical solution, at least for some of the issues to be addressed in opportunistic networking.

More generally my results deal with the feasibility of forwarding in opportunistic networks and their consequences require more attention. At least three different directions may be followed.

- First, it might be that reasoning with expected value of delay is not suitable, since the possible occurrence of a long delay is unavoidable, whatever forwarding algorithm is used. Applications for such networks should therefore be designed to cope with this aspect of opportunistic communication.
- Second, note that I did not model the general case where contact processes for a pair of nodes are heterogeneous or contain significant correlation. It is still possible that a finite expected delay exists in a more complex model that reproduces accurately the statistical properties of the datasets. This direction is appealing but it requires to remove one of the modeling assumptions that I have made and that is instrumental for most of the results currently known in this area. It also necessitates to design a forwarding algorithm that differentiates between nodes; some schemes of that type have been only recently proposed [DFGV03, LDS04, LFC06].

[10]It sounds bad, right? But actually for many applications, we care more about how many percentages of the messages can be delivered instead of the expected delay. It is just like best effort and QoS.

- Third, one can investigate how to add connection opportunities in a mobile network, using special devices or partial infrastructure, that could in some cases be already available. It looks promising but the impact of this partial infrastructure should be carefully studied.

Although the results showed in the chapter are not positive for forwarding in PSN, we still need to do forwarding. The two “oblivious” algorithms I introduced in this chapter provides two bounds for all kind of forwarding algorithms. If I do not consider wireless contention and loss rates, no algorithm can achieve better performance than flooding in terms of delivery delay. And in terms of delivery cost, wait-and-forward would be considered to be the best choice. But for overall performance, neither approach is ideal. Wait-and-forward is too slow and will have low delivery ratio. Flooding packets has a very high cost, not just in link utilisation, but for other resources such as node storage and battery life, which are likely to be highly valued by users. Controlled flooding is not only easy to implement but it can also be used to achieve several performance-cost objectives. Getting close to the optimal delay is possible with a small number of intermediate relays, but it remains costly with such simple algorithms. I envision that success delivery rate and cost could be further optimised using more sophisticated techniques.

In the research community, it has been a widely held belief that identifying community information about recipients can help select suitable forwarders, and reduce the delivery cost compared to “oblivious” flooding. This is a reasonable intuition, since people in the same community are likely to meet regularly, and hence be appropriate forwarders for messages destined for other members of their community. However, to date as far as I am aware, there has been no experimental evaluation of this belief, and no-one knows whether it is valid or not. In the rest of the dissertation, I will focus on exploring the possibility and effectiveness of using social information for forwarding. In the next chapter, I will first look at how can we infer human communities from the datasets.

Chapter 3

Inferring Human Communities

In last chapter, I have shown that human inter-contact time can be approximated by a power-law in the [10 minutes; 1 day] range, which gives us some rough idea about pair interaction. In order to understand more about human mobility and interaction, I want to look at group-level interaction in this chapter. I apply some complex network analyses such as K-CLIQUE community detection [PDFV05] and Newman's weighted network analysis (WNA) [J.N04] to the human mobility traces, which allow us to understand the human clustering behavior[1] in different environments and we can also use these detected communities for further study of social-aware forwarding[2]. For online applications, I also propose three distributed community detection algorithms. This is joint work with my supervisor Prof. Jon Crowcroft, Dr Eiko Yoneki and my fellow PhD student Shu-Yan Chan. Shu-Yan helped me to implement WNA in Java, and I coordinated and completed the remaining parts.

3.1 Introduction

A social network consists of a set of people forming socially meaningful relationships, where prominent patterns or information flow are observed. In PSN, social networks could map to computer networks since people carry the computer devices. The aims of this chapter are to uncover the social community structures from the mobility traces using centralised community detection methods, and to develop distributed version of detection algorithms for practical online applications.

Mobility traces can be represented in the form of weighted graphs called contact graphs, with the weight of an edge representing the contact duration/contact frequency of the two end vertices. Understanding human interaction can be tackled from the domain of weighted network analysis. One possible outcome of studies of the weighted contact graphs is community detection. Many

[1]This includes how people form cliques and interact with each other on a group basis.

[2]For K-CLIQUE detection, I apply the algorithm directly, but for WNA I find two different versions of the algorithm in the original paper and I need to choose the correct version and implement it.

real-life networks are weighted, but because of complexity, not a great deal of analysis has been done in this area. The seminal work is a WNA paper by Newman [J.N04]. The advantage about weighted analysis is that, unlike other algorithms, we do not need to threshold the weight on the contact graphs. The disadvantage is that it cannot detect overlapping community structures, yet in human society one person may belong to multiple communities. To address this problem, I also use the K-CLIQUE community algorithm proposed by Palla *et al.* [PDFV05], which allows overlapping of communities.

I evaluate the algorithms on mobility traces, which have *a priori* knowledge of community so that I can compare the detected communities to reality. Additionally, I evaluate the impact of the detected communities on message forwarding efficiency in Chapter 4 and Chapter 5, and I find out that such community information improves forwarding efficiency quite significantly.

As we are targeting online forwarding applications, I also sought distributed community detection algorithms which can allow the mobile devices to detect their own local communities instead of relying on a centralised server. Here I propose three algorithms, namely SIMPLE, K-CLIQUE, and MODULARITY, which are demonstrated to achieve quite close performance to the centralised methods, in the best case around 90% accuracy. [3]

The structure of this chapter is as follows. I survey existing community detection algorithms in Section 3.2, followed by the characteristics of contact graphs in Section 3.3. I go into the details of the methodologies, including WNA in Section 3.4 and K-CLIQUE community detection in Section 3.5. I present the algorithms on distributed community detection and the results in Section 3.6 and Section 3.7. Finally I conclude with a brief discussion.

3.2 Community Detection

Community detection in complex networks has attracted a lot of attention in recent years. There is still no universally accepted definition of community, but in most versions, community is a subgraph of a network whose nodes are more tightly connected with each other than with nodes outside the subgraph. Detecting community is equivalent to investigating statistical properties of a graph, disregarding the roles played by specific subgraphs, and hence identifying substructures/subgraphs which could correspond to important functions. In the case of the World Wide Web, examples of communities are sets of Web pages dealing with the same topic [FLGC02]. In biological networks, it is widely believed that modular structure results from evolutionary constraints and plays a crucial role in biological functions [HHLM99] [RSM$^+$02]. In social networks, community structures correspond to human social communities [New04b] [LN04]. Finally on the Internet, community structures correspond to autonomous systems [LN04], which are a connected segment of a network consisting of a collection of subnetworks interconnected

[3]The whole trace is used for the training in order to compare with the centralised methods, which use the whole trace for the calculation.

by a set of routers. In the PSNs I studied, community structure would correspond to human communities or some structures which are beneficial for forwarding efficiency. Given the relevance of the problem, it is crucial to construct efficient procedures and algorithms for the identification of the community structure in a generic network. Recent reviews [New04b] and [DDDGA05] may serve as introductory reading, which also include methodological overviews and comparative studies of the performance of different algorithms.

Qualitatively, a community is defined as a subset of nodes within the graph such that connections between the nodes are denser than connections with the rest of the network. The detection of community structure in a network is generally intended as a procedure for mapping the network into a tree [RCC+04], known in social science as dendrogram. In this tree, the leaves are the nodes whereas the branches join nodes or (at higher level) groups of nodes, thus identifying a hierarchical structure of communities nested within each other.

There are two main approaches to clustering: nodes are either joined successively in an agglomerative manner starting from single nodes, or the whole network is recursively partitioned. Hierarchical clustering is a representative of the agglomerative approach [WFI94] . Using this method, nodes are grouped into larger and larger communities, and the tree is built up to the root, which represents the whole network. For the divisive approach, the order of construction of the tree is reversed: one starts with the whole graph and iteratively cuts the edges, thus dividing the network progressively into smaller and smaller disconnected subnetworks identified as the communities. The crucial step in a divisive algorithm is the selection of the edges to be cut. Girvan and Newman (GN) have introduced a divisive algorithm in which the selection of the edges to be cut is based on the value of their edge betweenness [NG04], a generalisation of Freeman betweenness centrality [Fre77]. The betweenness of an edge is the number of shortest paths between all node pairs running through it. It is clear that, when a graph is made of tightly bound clusters, each loosely interconnected, all shortest paths between nodes in different clusters have to go through the few inter-cluster connections, which therefore have a large betweenness value. Recursively removing these large betweenness edges would partition the network into communities of different sizes. The GN algorithm represents a major step forward for the detection of communities in networks, since it avoids many of the shortcomings of traditional methods [NG04].

Quantitatively, however, we need metrics to measure how well the community splitting is progressing, otherwise most of the algorithms would continue until every node is split into a single community. Newman and Girvan proposed in [NG04] a measure of networks called *modularity*. Such as for a division with g groups, they define a $g \times g$ matrix **e** whose elements e_{ij} are the fractions of edges in the original network that connect vertices in group i to those in group j. The modularity is defined to be

$$Q = \sum_i e_{ii} - \sum_{ijk} e_{ij} e_{ki} = \mathtt{Tr}\,\mathbf{e} - \|\mathbf{e}^2\| \tag{3.1}$$

where $\|e^2\|$ indicates the sum of all elements of e^2, and $\mathtt{Tr}\,\mathbf{e}$ is the trace of matrix **e**, which is the sum of the diagonal elements. This measure essentially compares the number of links inside a given module with the expected value for a randomised graph of the same size and same degree sequence. The concept of modularity has gained such popularity that it has not only been used as a measure of the community partitioning of a network; it has also been used as a heuristic indicator in various community detection algorithms [NG04] and also as the sole quality or fitness function in community detection algorithms [CNM04]. I will revisit the concepts of modularity in the next section.

Neither the agglomerative nor the divisive methods consider the overlapping of communities, but in nature many nodes may belong to several communities at the same time, just as a human in the society may belong to many different social groups. Palla *et al.* define a k-clique community as a union of all k-cliques (complete subgraphs of size k) that can be reached from each other through a series of adjacent k-cliques [PDFV05]. Two k-cliques are said to be adjacent if they share $k-1$ nodes. This definition is based on their observation that an essential feature of a community is that its members can be reached through well-connected subsets of nodes, and that there could be other parts of the whole network that are not reachable from a particular k-clique, but they potentially contain further k-clique communities. Or in other words, the k-clique communities of a network with $k = 2$ are equivalent to the connected components, since a 2-clique is simply an edge and a 2-clique-community is the union of those edges that can be reached from each other through a series of shared nodes. On the another hand, a 3-clique-community is the union of triangles that can be reached from one another through a series of shared edges. As k is increased, the k-clique communities shrink in size, but become more cohesive since their member nodes have to be part of at least one k-clique.

Most of the algorithms mentioned above are dealing with binary graphs, which are undirected and unweighted. There are many everyday examples of networks, such as the Internet, the world wide web, and various biological and social systems. Many of these are intrinsically weighted, their edges having differing strengths, e.g. in a social network there may be stronger or weaker social ties between individuals. However, there are so many cases where edge weights are known for networks, and to ignore them is to throw away a lot of data that, in theory at least, could help us to understand these systems better. As I have introduced in Section 3.1, Newman [J.N04] proposed a transformation of the edge-betweenness community detection algorithm from an unweighted network to a weighted version. I will discuss the details of this algorithm in later sections.

3.3 Contact Graphs

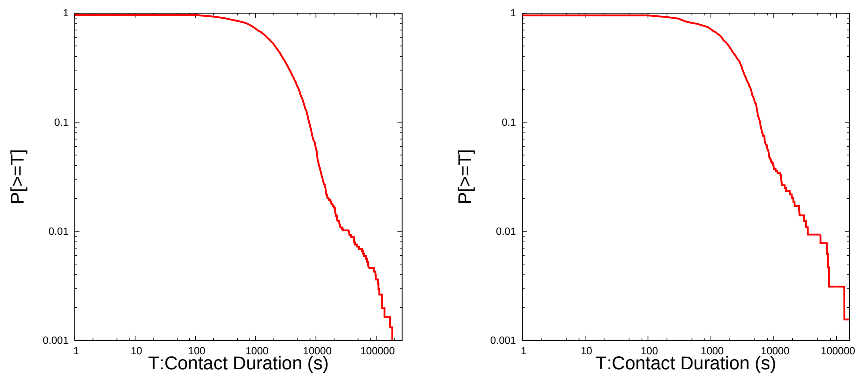
My first contribution is to introduce the notion of *contact graph* as a way to help represent the mobility traces, and to choose a threshold for community detection. The way I convert human mobility traces into weighted contact graphs is based on the number of contacts and the contact duration, although I could use other metrics. The nodes of the graphs are the physical nodes from the traces, the edges are the contacts, and the weights of the edges are the values based on the metrics specified such as the number of contacts during the experiment.

We can measure the relationship between two people by how many times they meet each other and how long they stay with each other. We naturally think that if two people spend more time together or see each other more often, they are in a closer relationship. In this chapter I am not going to provide a specific threshold to infer actual social context. I just use these two metrics to produce maps which may prove useful to guide forwarding, although later we will find out that the detected communities match well with the real social communities.

Here I explore further properties of the experimental scenarios, and present statistics concerning the contact graphs for each dataset.

3.3.1 Weight Distribution of Contact Graphs

First I show that the statistical properties for the two conference scenarios are quite similar. Figure 3.1(a) and 3.1(b) show the contact duration distribution for *Infocom06* and *Infocom05*



(a) *Infocom06* (b) *Infocom05*

Figure 3.1: Contact duration distribution for *Infocom06* and *Infocom05*

respectively. The x-axis is the contact duration in seconds and the y-axis is the probability of contact durations larger than the value on the x-axis. We can see that their distributions are quite similar, with a mean difference as small as 0.0003 (0, 0.0633). More similarities will be seen in the next section as well. To prevent redundancies, the later sections I only selectively show one example, in most cases *Infocom06*, since it contains more participants.

Figure 3.2 and Figure 3.3 show the contact duration and number of contacts distribution for each pair in four experiments. For the *HongKong* experiment I include the external devices, but for the other three experiments I use only the internal devices. I show later that for the *HongKong* experiment I need to use the external devices to help to forward the data because of network sparseness.

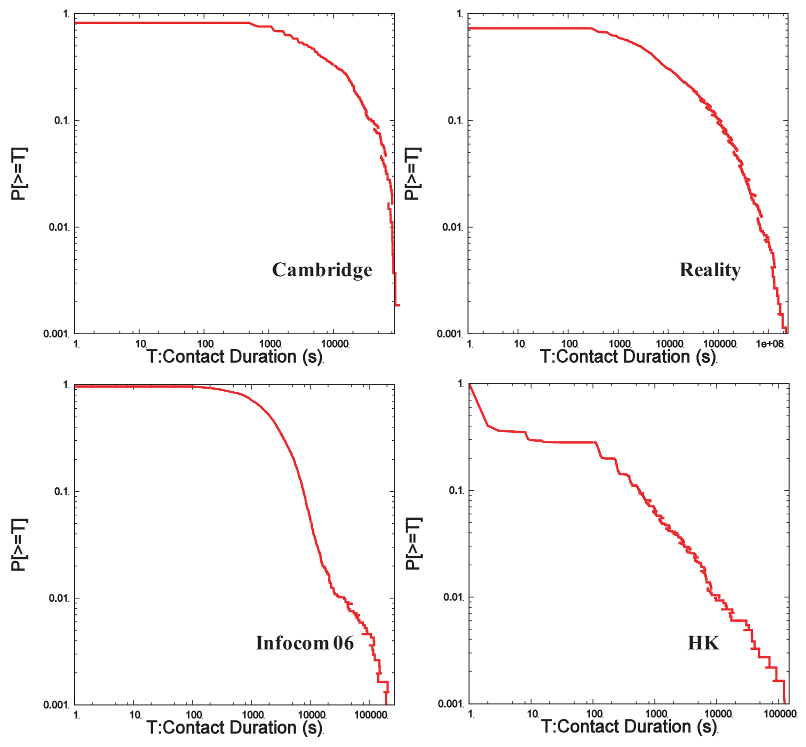


Figure 3.2: The contact duration distribution for each pair in four experiments.

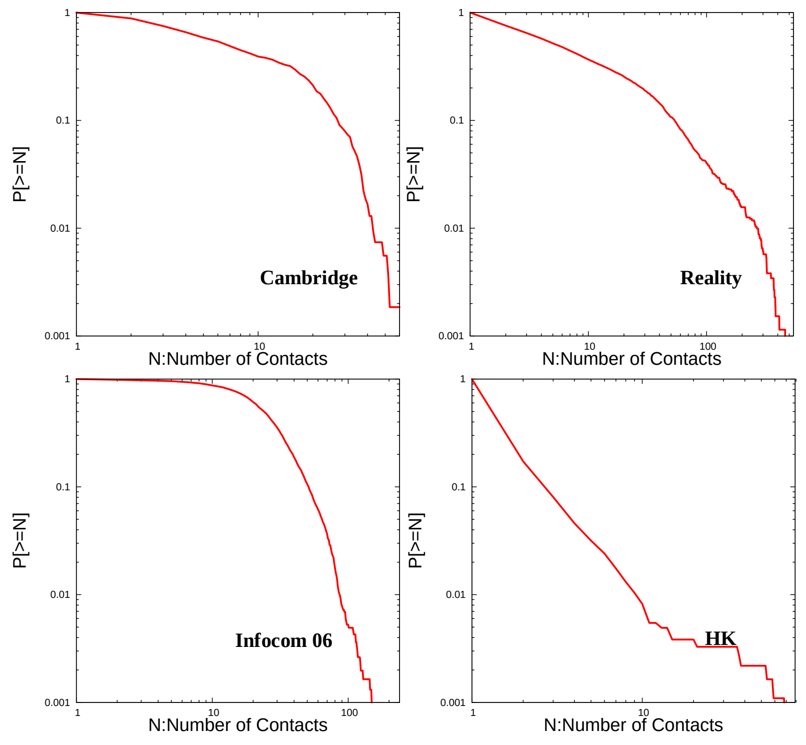


Figure 3.3: The number of contacts distribution for each pair in four experiments.

3.3.2 Correlation between Regularity and Familiarity

I assume contact duration indicates familiarity. Two people sharing the same office might hate each other, and not talk, but I will ignore this kind of extreme situation here. The number of times two people meet each other implicitly reveals the pattern with which they meet. In this work, I infer regularity of meetings from the number of contacts. Two people might meet a lot of times in a short period (e.g. a day), and then not at all. However, short periods with many contacts are less likely to contribute to the upper quarters of the distribution, and here I will ignore these too as outliers.

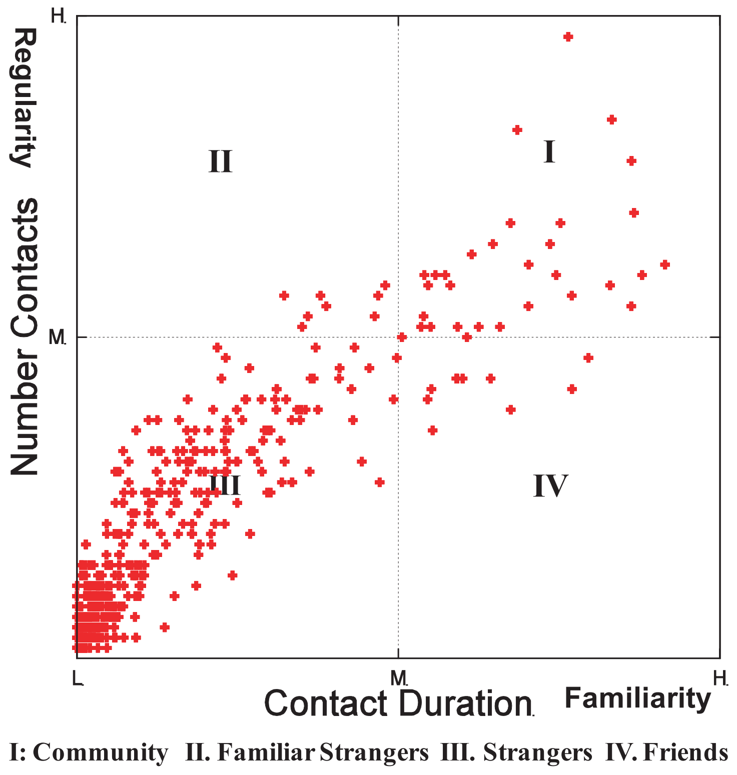


Figure 3.4: Number of contacts versus the contact durations for pairs of *Cambridge* Students.

Figure 3.4 shows the correlation between regularity and familiarity in the *Cambridge* dataset[4]. Here regularity is positively correlated to familiarity with a correlation coefficient of 0.9026. I define four kinds of relationships between a pair of nodes: Community, Familiar Strangers, Strangers, and Friends. A pair of nodes which has long contact duration (high familiarity) and large number of contacts (high regularity) is likely to belong to the same community. A pair of nodes which meet regularly but do not spend time with each other, could be familiar strangers [PG04] meeting everyday. People who do not meet regularly and do not spend time with each other would be in the category of strangers. Finally, for node pairs which do not meet very frequently but spend quite a lot of time together for each meeting, I count as friends. It

[4]Here, by *Cambridge* data I refer to the *Cambrige05* data in Section2.3. The details of other datasets can also be found in this section.

is not necessary that the division of the four quarters are exactly at the middle. It is just as a reference or example. A clear-cut division may need more empirical experimental results. But here I provide the methodology to classify these four kinds of relationship based on pure contact duration and frequency. Additional difficulties faced by empirical social network research are well described in work by Watts [Wat99].

Figure 3.5 shows the correlation between the number of contacts and contact durations for the other four experiments. We can see that conference environments are quite similar, both with a narrow stripe in the left bottom quarter. This stripe shows that people in the conference tend to meet each other more often than spend long time together. That is a typical conference scenario, since people may meet each other many times in coffee breaks, corridors, the registration desk etc. They may stand together and chat for a while, and then shift to chat with others instead of spending all the time together. *Infocom06* contains double the number of participants, and hence more data points. The *Reality* set is similar to the *Cambridge* one, with most of the points lying on or above the diagonal line. However, it seems that people also have more contacts instead of spending time together. In the *HongKong* figure, we can find two pairs of friends, two pairs of close community members, and two pairs of familiar strangers. All the other pairs lie in the strangers quarter. This is in line with our expectations for an experiment designed to contain little social correlation.

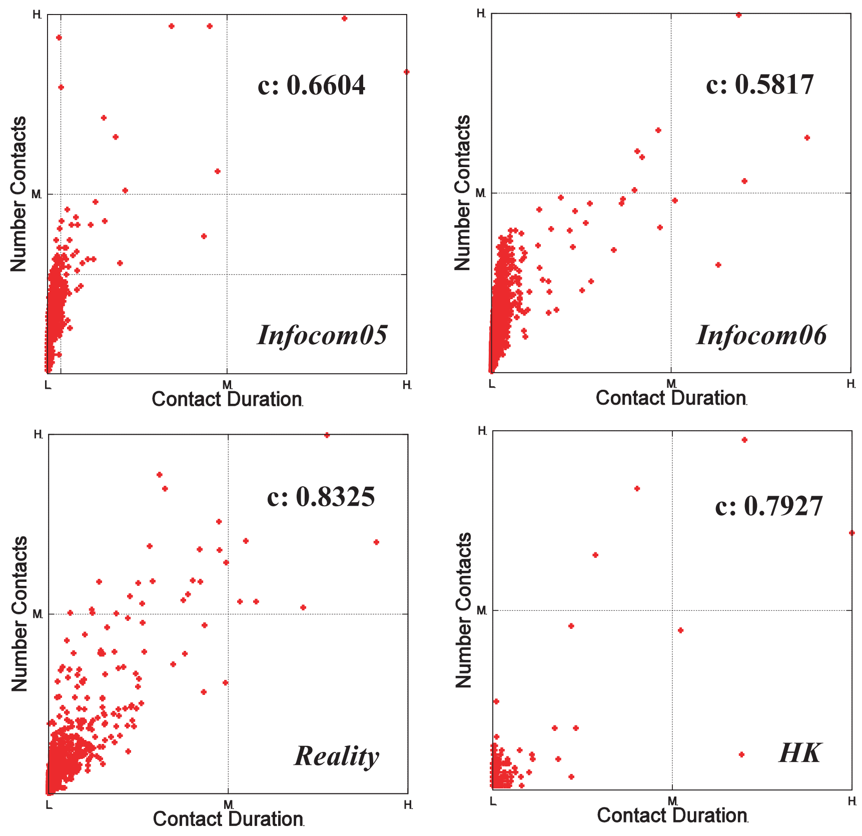


Figure 3.5: Number of contacts against contact durations for all pairs in the four datasets, with correlation coefficient.

These correlation graphs give us an overview idea about the social characteristics of the people in these environments. And later in the community detection part, I will detect different types of relationships by choosing suitable thresholds for[5] the contact duration and number of contact similar to the four quadrants division in Figure 3.4.

3.4 Weighted Network Analysis

In this section, I implement and use Newman's WNA for data analysis, even though the algorithm was proposed two years ago and yet no reference implementation publicly exists. [6] And I also extend unweighed *modularity* proposed in [NG04] to a weighted version and use it as a measurement of the fitness of the detected communities. I will start from the *modularity* and then move to the *betweenness* algorithm.

3.4.1 Clustering by Modularity

As mentioned previously, Newman and Girvan proposed in [NG04] a measure of network cohesiveness called *modularity*. The concept of modularity has gained such popularity that it has not only been used as a measure of community partitioning of a network, it has also been used as a heuristic indicator guiding various community detection algorithms [NG04], and also used as the sole quality or fitness function in community detection algorithms [CNM04].

For each community partitioning of a network, one can compute the corresponding modularity value using the following definition of *modularity* Q:

$$Q = \sum_{vw}[\frac{A_{vw}}{2m} - \frac{k_v k_w}{(2m)^2}]\delta(c_v,c_w) \tag{3.2}$$

where A_{vw} has value 1 if vertices v and w are connected and 0 otherwise, $m = \frac{1}{2}\sum_{vw} A_{vw}$, c_i is the index of vertex i, and hence $\delta(c_v,c_w) = 1$ iff vertices v and w belong to the same community, and $= 0$ otherwise. Therefore the term in the formula $\frac{\sum_{vw} A_{vw}}{2m}\delta(c_v,c_w)$ is equal to $\frac{\sum_{vw} A_{vw}\delta(c_v,c_w)}{\sum_{vw} A_{vw}}$, which is the fraction of the edges that fall within communities. Modularity is defined as the difference between this fraction and the fraction of the edges that would be expected to fall within the communities if the edges were assigned randomly but keeping the degrees of the vertices unchanged. Under such a condition of random edge allocation, the probability for a given edge connected to vertex v is $\frac{k_v}{2m}$ where k_v is the degree of vertex v defined as $\sum_w A_{vw}$. It follows that the probability for a given edge connected to vertex v and w is $\frac{k_v k_w}{(2m)^2}$. Summation of this term over all v and w results in the fraction of the edges that would

[5]Many community detection algorithms only work on binary graphs, and we need to convert a weighted graph into a unweighted graph by setting a threshold.

[6]During the implementation, I found two different interpretations of the algorithm in the paper. I believe that I have chosen the correct one after the confirmation of the author.

expected to fall within the communities if the edges were assigned randomly, and hence forms the term used in the above formula for Q.

In [CNM04], Clauset *et al.* proposed an optimised algorithm to detect community structure of unweighed graph, based solely on the concept of modularity using an agglomerative approach. It first assigns each individual node to its own community, then chooses the two communities to merge together which would give the maximum increase in the modularity value of the community partitioning.

As stated in [J.N04] according to the "general" transformation of algorithms for unweighted networks, the modularity Q for weighted networks has the same formula

$$Q = \sum_{vw}[\frac{A_{vw}}{2m} - \frac{k_v k_w}{(2m)^2}]\delta(c_v,c_w) \tag{3.3}$$

provided A_{vw} now represents the weight of the edge between v and w, and the degree k_i is defined to be $\sum_w A_{vw}$ as before .

However, instead of calculating the adjacency matrix of the graph and calculating at each step the possible changes in Q due to further merging ($\triangle Q_{ij}$), the optimised (unweighed) algorithm in [CNM04] maintains and updates a sparse matrix containing $\triangle Q_{ij}$ for each pair i,j of communities with at least one edge between them. In [CNM04], the initial value for $\triangle Q_{ij}$ is set to

$$\triangle Q_{ij} = \frac{1}{2m} - \frac{k_i k_j}{(2m)^2} \tag{3.4}$$

if i,j are connected, and $Q_{ij} = 0$ otherwise (in these cases, Q_{ij} is set to 0 as an optimisation since joining two communities with no edge between them can never produce an increase in Q, so we do not need to consider joining them). At each iteration, the two communities i and j with the largest $\triangle Q_{ij}$ are chosen to merge together, until only one community remains. The $\triangle Q_{ij}$ value is updated according to the rule set. The weighted version of the algorithm and program remain the same efficiency as the original version: $O(mdLog(n))$ for a network that has m edges, n vertices, and where d is the depth of the "dendrogram" describing the network's community structure. In this case, m is the number of edges in the weighted version of the graph, not the number of edges in the transformed multigraph).

I observe that the original formula purposed in [CNM04] has a mistake in it. There is a factor of 2 missing in the initial value of $\triangle Q_{ij}$. The correct formula should be:

1. $$\triangle Q_{ij} = \frac{\mathbf{2}}{2m} - \frac{\mathbf{2}k_i k_j}{(2m)^2} \tag{3.5}$$

   if i,j are connected, and $\triangle Q_{ij} = 0$ otherwise, for unweighted networks.

2. $$\triangle Q_{ij} = \frac{\mathbf{2}A_{ij}}{2m} - \frac{\mathbf{2}k_i k_j}{(2m)^2} \tag{3.6}$$

   if i,j are connected, and $\triangle Q_{ij} = 0$ otherwise, for weighted networks.

For both the weighted and unweighted version, if vertices i and j are merged, the difference in the $-\frac{k_v k_w}{(2m)^2}\delta(c_v,c_w)$ term in the modularity value before and after the merge is $-\frac{\mathbf{2}k_i k_j}{(2m)^2}$ since after the merge, $\delta(c_v,c_w) = 1$ for $v = i, w = j$ or $v = j, w = i$.

The $\frac{1}{2m}$ term in the definition for $\triangle Q_{ij}$ in [CNM04] is derived from the $\sum_{vw} \frac{A_{vw}}{2m}\delta(c_v,c_w)$ term in the original definition of Q, of which after vertices i and j were merged to a single community, $\delta(c_v,c_w) = 1$ for $v = i, w = j$ or $v = j, w = i$, so the new Q value should include the new term $\frac{A_{vw}}{2m}$ and $\frac{A_{wv}}{2m}$ in the summation, and for a unweighted network with no parallel edges, A_{vw} and A_{wv} both only equal 1. Therefore, for the unweighted version of the algorithm, the correct term should be $\frac{\mathbf{2}}{2m}$ and for the weighted version, $\frac{\mathbf{2}A_{ij}}{2m}$.

The algorithm is essentially a genetic algorithm in disguise, using modularity as the measure of fitness. Instead of testing some mutations of the current best solutions, it enumerates all possible mergings of any two communities in the current solution, and evaluates the relative fitnesses of the resulting merges. The merge is considered to be fit if it leads to an increase in modularity value. The algorithm proceeds with the merge which gives the highest increase in modularity value as the next current solution, and terminates if no possible merge would increase the modularity value.

In this chapter, I use a community detection algorithm based on edge betweenness, which I will talk about in following section, and which uses modularity to evaluate the fitness of a particular division.

3.4.2 Clustering by Edge Betweenness

As has also been briefly introduced before, Newman and Girvan [NG04] proposed a community detection algorithm based on edge betweenness for unweighted networks. They defined the *edge betweenness* of an edge in a network as the number of geodesic (shortest) paths between all node pairs that run along that edge. If there are two geodesic paths joining a given node pair, then each one counts as a half of a path, and similarly for three or more. At each iteration of the community detection algorithm, edge betweenness of all edges are (re-)computed, then the edge with the highest edge betweenness is then removed. An existing community may then be split into two because the removed edge might be the sole connection between these two communities (the reason behind this algorithm is that the inter-community edges will be the ones visited most by the geodesic path). Eventual all nodes are split into their own communities. Then the modularity values of the community partitioning at each iteration are computed. According to Newman and Girvan [NG04], the local peaks of the modularity value correspond to “satisfactory” splits.

Although the impression given in [J.N04] is that they obtain their weighted version of the algorithm by following the general mapping of a weighted network to a multigraph with multiple edges of unit length, there are actually two completely different interpretations in the paper. I call them *Interpretation Bad* and *Interpretation Final*, respectively.

Interpretation bad : The author stated “To derive an answer we employ our mapping from the weighted network to a multigraph. Suppose we have a weighted network with integer weights on the edges and as before we replace each edge of weight n by n parallel edges of unit weight. The adjacency matrix remains unchanged. Now we apply the normal unweighted version of our algorithm to the resulting multigraph.”

Interpretation Final: In the same paper, Newman also summarised the adaptation of this algorithm (which itself is based on Dijkstra’s shortest path) for a weighted network as following: “first calculate the betweennesses of all edges in a weighted network ignoring the weights. Then divide each such betweenness by the weight of the corresponding edge, remove the edge(s) with the highest resulting score, recalculate the betweennesses, and repeat.”

Interpretation Bad is only equivalent to *Interpretation Final* for graphs which have shortest paths without cycles. I provide the following example to illustrate the different results these contradictory interpretations will produce:

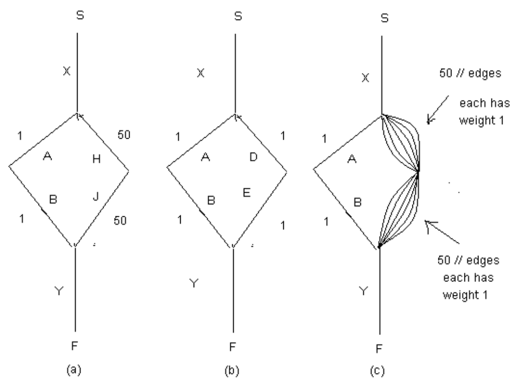


Figure 3.6: Illustration of Newman edge betweenness

In the network given in Figure 3.6(a), if we want to compute the edge betweenness, then according to *Interpretation Final*, we will calculate the betweennesses of all edges in our weighted graph in the normal way, ignoring the weights. So if we consider the contribution of the shortest path from S to F of the network, we first compute its contribution to the transformed network as shown in Figure 3.6(b). There are 2 shortest paths from S to F, and their contributions to the betweenness of each of the edges A,B,D,E are 1/2. Then the next step is to divide each such betweenness by the weight of the corresponding edge. Therefore, the betweenness values of the edges are now: $A = B = 1/2$, $D = E = 1/(2 \times 50) = 1/100$.

However, if we follow *Interpretation Bad*, we will replace each edge of weight n by n parallel edges of unit weight as shown in Figure 3.6(c). Then we apply the normal unweighted version of our algorithm to the resulting multigraph. When traveling from S to F in the network, there will be $1 + 50 \times 50$ equal-length shortest paths, so the contribution to the betweenness value by edges A and B will be $1/2501$, but the contribution by each of the 50 parallel edges will be $50/2501$.

As we can now see, the ratio of the contribution of the same path (S to F) to the edges' betweenness values are quite different according to the different interpretations of the algorithm. For *Interpretation Bad* the ratio of betweenness contribution of A to H (the original edge with weight 50) $= 1/2501 : 50/2501 = 1 : 50$. For *Interpretation Final*, the ratio $= 1/2 : 1/100 = 50 : 1$ The appearance of 50 in both ratios are coincidental. If we change the original weight of edge J to 30, then the calculation for *Interpretation Bad* will become $1/1501 : 30/1501 = 1 : 30$ while *Interpretation Final* will give the ratio $= 1/2 : 1/(2 \times 50) = 50 : 1$.

Interpretation Final can be restructured into the follow generic structure, and is the version that our further discussion of community detection on weighted networks will be based on:

1. For each pair of vertices in the network, compute the "shortest path" between them according to some "shortest path" measurement (for *Interpretation Final*, the path length is equal to the number of hops along the path), accumulate the number of "shortest paths" passing through each edge and repeat for the other pairs. If there are multiple "shortest paths", their contributions to the accumulated values of any particular edge is the fraction of the multiple paths passing through that edge.
2. The edge betweennesses are then the values accumulated for each edge divided by their edge weights.
3. Remove the edge with the highest edge betweenness. and repeat from 1 until there are no more edges in the network.
4. For each step in 3 that splits a community into two smaller ones, recalculate the modularity value of the network with the current community partitioning. Select those splittings with local maxima of modularity.

As it can now be seen in step 2 above, along any "shortest path", it penalises heavy edges' (e.g. edges that indicate high proximity) shares of the contribution of that path to their final edge betweenness. The aim of this step is to force the inter-community links along the shortest path to be selected by the algorithm to have the highest edge betweenness, as Newman suggests in [J.N04] that the inter-community edges should always have lower weight then those intra-community edges.

In this chapter, I will use the above version of weighted network analysis and also the K-CLIQUE detection algorithm to study the human interactions in the datasets. They both have advantages and disadvantages, but together they are very useful tools for our study. In the following subsection, I will first give the results by the weighted analysis.

3.4.3 Results by Weighted Analysis

As I mentioned before, the traces are converted into weighted contact graphs. In this chapter, I create contact graph based only on two metrics: one is the number of contacts and the other one is contact duration. These contact graphs are fed directly into the algorithm for community detection.

Table 3.1 summarises the communities detected using Newman's algorithm on all three datasets. Nonzero Q values indicate deviations from randomness; values around 0.3 or more usually indicate good divisions. For the *Infocom06* case, the Q_{max} value is low. This indicates that the community partition is not very good in this case; this also agrees with the fact that in a conference, the community boundary becomes blurred (people with different affiliations mix with each other). For the *Reality* case, the Q value is high. This also reflects the more diverse campus environment. Out of the 100 participants, 75 are either students or faculty in the MIT Media Laboratory, while the remaining 25 are incoming students at the adjacent MIT Sloan Business School. Of the 75 users at the Media Lab, 20 are incoming masters students and 5 are incoming MIT freshmen. For the *Cambridge* data, the two groups split by the algorithm exactly match the two groups (2nd year and 3rd year) of students selected for the experiment. The scenario of *Infocom05* is similar to *Infocom06* so I will not go into the details. I do not want to claim that the communities detected by the algorithm exactly match the real social communities, but at least reflect some of the social relationships. In my study here, my main focus is to see how these detected communities have impact on forwarding efficiency.

Experimental dataset	Infocom06	Cambridge	Reality	Infocom05
Q_{max}	0.2280	0.4227	0.5682	0.3039
Max. Community Size	13	18	23	13
No. Communities	4	2	8	4
Avg. Community Size	8.000	16.500	9.875	6.5
No. Community Nodes	32	33	73	26
Total No. of Nodes	78	36	97	37

Table 3.1: Communities detected from five iMote datasets

To prevent redundancy, I will not provide visualisation of the WNA communities detected. Instead I will show that for the K-clique communities. It is more interesting to show the overlapping of community structures than only showing the stand-alone version.

3.5 Finding k-clique Communities

I use the K-CLIQUE community algorithm proposed by Palla *et al.* [PDFV05], since overlapping of communities is allowed, and it is apparent that in human society one person may belong to multiple communities. I have calculated all the results by using both contact duration and number of contacts for all five experiments but to prevent redundancies I just show two cases of contact duration and two cases of number of contacts.

I will show both graphically and quantitatively the overlapping communities within all these environments in the following.

3.5.1 k-clique University Communities

In the visualisation, an edge is added between two nodes if they are direct neighbors to each other in the community. The length of the edges is not proportional to any property of either the communities or the nodes. However the width of the edges is proportional to the link-weight that is the number of shared nodes between the two communities.

Figure 3.7 shows the k-clique communities detected from the *Cambridge* data using number of contacts.

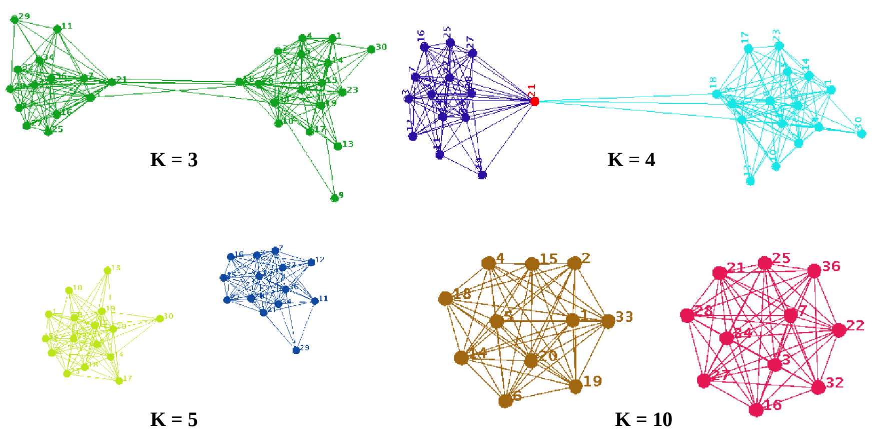


Figure 3.7: Communities based on number of contacts with weight threshold =29, k = 3,4,5, and 10 (*Cambridge*).

The duration of the experiment is 11 days. For the number of contacts, I used a threshold of 29 contacts, which represents an average of almost 3 contacts per day.[7] In this case, around 8.5% of all the edges are taken into account. I observe that the nodes mainly split into two communities of size 11 respectively with k as high as 10. Next I examine lower values of k. We can see also from Figure 3.7, when $k = 3$ there is a big community of 31 nodes, and when $k = 4$ the big community splits into two overlapping communities of sizes 14 and 17 with overlapping size of 1, and when $k = 5$ the two overlapping communities split into two disjoint communities of size 14 and 16 respectively. The two disjoint community structures are visible until $k = 11$, with a gradual decrease in the community size. For the contact duration metric, I set the contact duration threshold to be 10 hours for the whole 11 days of the experiment (Figure 3.8). I also mainly observe two communities when using this metric. The membership of these two communities is more or less the same as that when using the number of contacts metric. This agrees with Figure 3.4 where we can observe that the contact duration and number of contacts for the *Cambridge* data are highly correlated.

[7]Considering some students may be taking the same courses, be in the same supervision group, and live in the same College, and hence using the same dining hall, this value is reasonable.

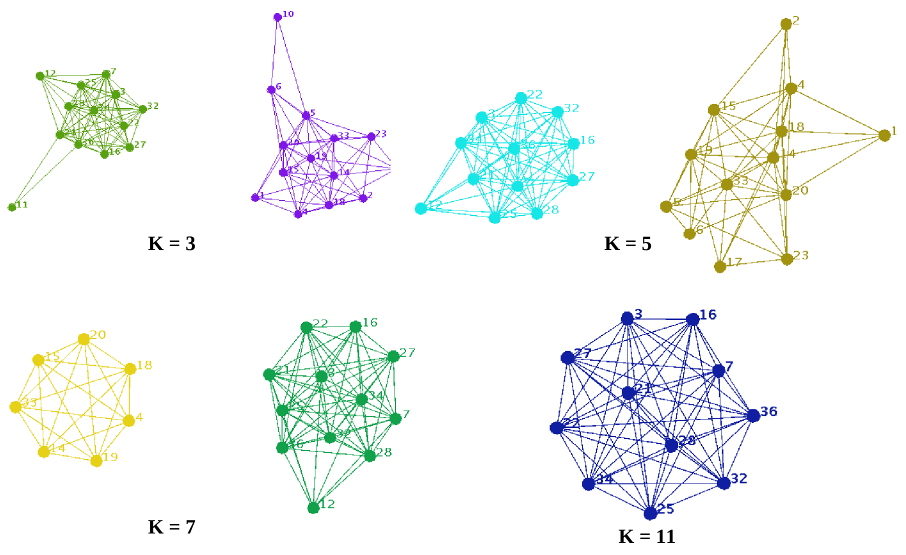


Figure 3.8: Communities based on contact durations with weight threshold = 10 hours, $k = 3,5,7,$ and 11 (*Cambridge*).

The output from the algorithm clearly illustrates that the participants can be seen as two communities in this case. When we look at the experimental data, the two communities classified by this algorithm match well with the two groups of Year1 and Year2 students selected for the experiment. Of course, in each group students tend to know each other and meet each other, and hence the clique size can be as large as 10.

3.5.2 k-clique Communities in Reality Mining

This is another campus environment but the environment is more diverse than the *Cambridge* one. We will see that unlike the *Cambridge* data, which consists mainly of two classes of students, this dataset consists of more groups.

First I look at communities detected by using a threshold of 388,800 seconds or 4.5 days on the 9-month *Reality* dataset. Here I assume 3 lectures per week and 4 weeks per month and for a total of 9 months, I get this threshold value (2% of the total links are taken into consideration). Research students in the same office may stay together all day so their contact duration threshold could be very large. For students attending lectures, this estimate can be reasonable. A looser threshold still detects the links with much stronger fit. I observe 8 communities of size (16, 7, 7, 7, 6, 5, 4, 3) when $k = 3$ in this case. The 4-size one overlaps at one node with the 16-size one which also overlaps with another 7-size community at another nodes. Two other 7-size communities overlap each other with overlapping size 1. The other three communities are disjoint. When $k = 4$, the 3-clique community is eliminated and other communities shrink or are eliminated, and only 5 communities of size (13, 7, 5, 5, 4) left. All of these 5 communities are disjoint. When $k = 5$, 3 communities of sizes (9, 6, 5) remain. The 9-size one and the 5-size

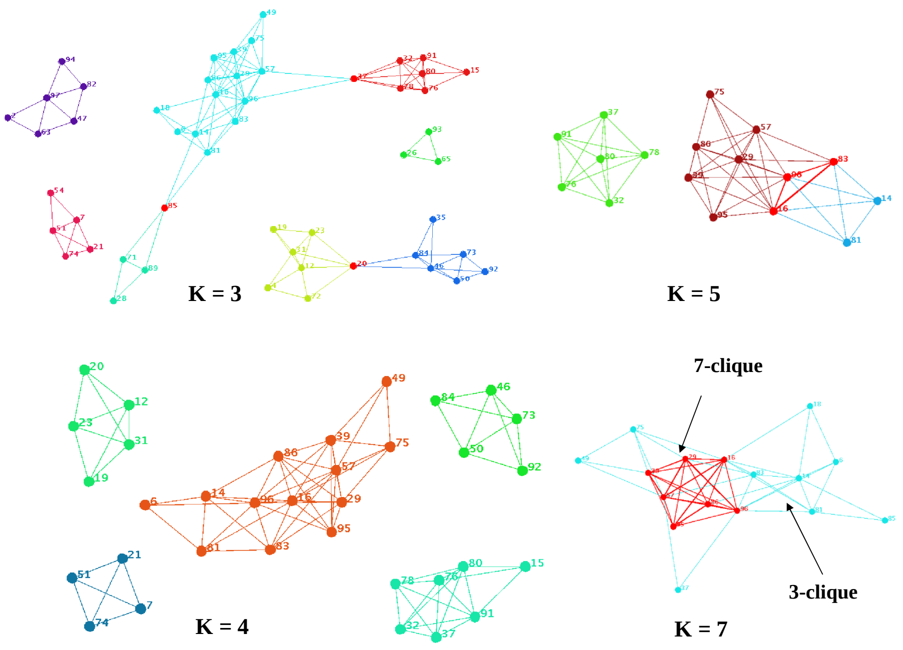


Figure 3.9: Communities based on contact durations with weight threshold = 388,800 seconds (4.5 days), $k = 3{,}4{,}5, and 7$ (*Reality*).

one are split from the 13-size one in the 4-clique case. Moving to $k = 6$ and $k = 7$, there are 2 communities and 1 community respectively.

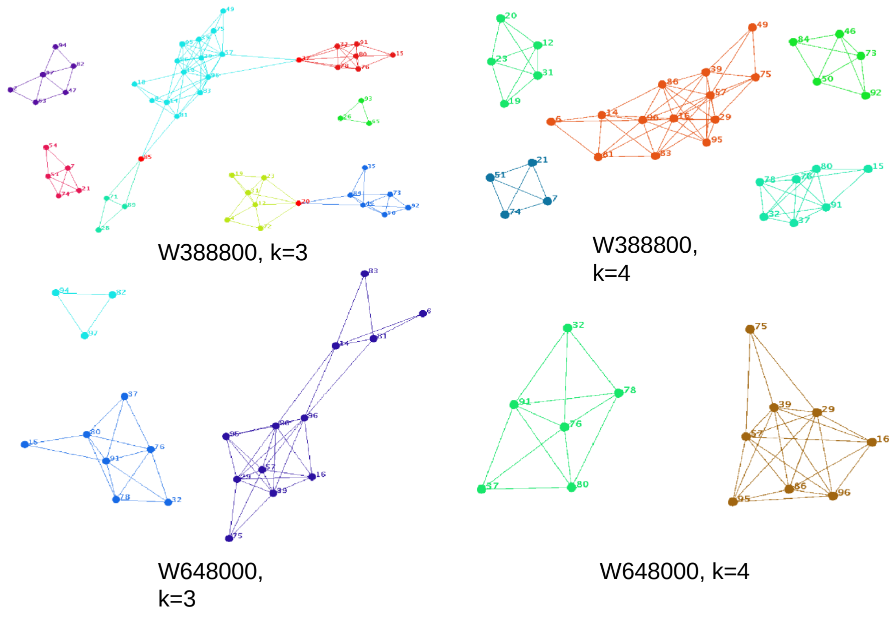


Figure 3.10: Communities based on contact durations with weight threshold = 648,000 seconds (7.5 days), $k = 3{,}4$ (*Reality*).

I am also interested in knowing about small groups which are tightly knit. I set a strict threshold of 648,000 seconds, that is on average 1 hour per weekday, 4 weeks per month, and for a total of 9 months. Around 1% of the links are taken into account for the community detection. When $k = 3$, there are three disjoint communities of size (12, 7, 3). When $k = 4$, there are only two communities left of size (8, 6). Figure 3.10 shows the 3-clique and 4-clique communities with 648,000-second threshold and 388,800-second threshold. A single 7-size community remains in the $k = 5$ and $k = 6$ cases. This 7-clique community is the same as in the 388,800-second case. These 7 people could be people from the same research group; they know each other and have long contact with each other.

3.5.3 k-clique Conference Communities

In this section, I will show the community structures in a conference environment. Here I take *Infocom06* as an example since it contains more participants than *Infocom05* and I have more participant information. Infocom is a multiple-track conference with several programs running at the same time. I do not expect all the 80 experiment participants to attend the same sessions, so will not expect the clique size to be as big as that of the *Cambridge* data. The total dataset only covers 3 days, hence I will not expect the threshold to be very big. People usually socialise during conferences in small groups, so I expect clique sizes of 3, 4 or 5 to be reasonable. And for *Infocom06*, the participants were specially selected so that 34 out of 80 form four subgroups according to academic affiliations. Out of these four groups, there were two groups from institutes in Paris with sizes of four and ten, respectively (named Paris Group A and Paris Group B), and there is one group from Lausanne, Switzerland of five people, and another, larger group of 15 people from the local organisation in Barcelona. But for this local organisation group, the volunteers are from different local institutions and also responsible for different sessions in the conference, so I will not expect them to all be together. After collecting the data, for privacy purposes, all the personal information about the participants is deleted except the Node ID, the affiliation and the nationality.

Figure 3.11 shows the 3-clique communities with threshold 20,000 seconds or 5.6 hours, that is approximately 1.85 hours per day. 1.68% of all edges are taken into account for the community calculation. I observe 6 communities of size (25, 11, 6, 6, 5, 3) in this case. The 25-size one overlaps at one node with a 6-size one which also overlaps with the 11-size community at another node and the 3-size one at another node. The 2nd 6-size community also overlaps the 3-size and 11-size one at another two nodes. The 5-size community stands alone. Although I know that during a conference people from different sub-communities tend to mix together and hence the boundary of affiliation communities should become less clear, I still find hints of the original affiliation communities from the figure. The algorithm correctly classified the nodes belonging to the local organisers into a community (see the Barcelona Group at the right hand side of the figure), and the members of the Lausanne Group into another community. There are several nodes which do not belong to these affiliations and are also falsely classified into the same communities, but this also truly reflects the nature of a conference, to socialise with people in other institutions. The two Paris groups are also clearly identified; they tend to socialise with each other. Node 47 belongs to both groups, it is important to link these two groups together. There are many members in the 25-size group not belonging to a common institution but they are here linked together by different small groups mixing together in the conference.

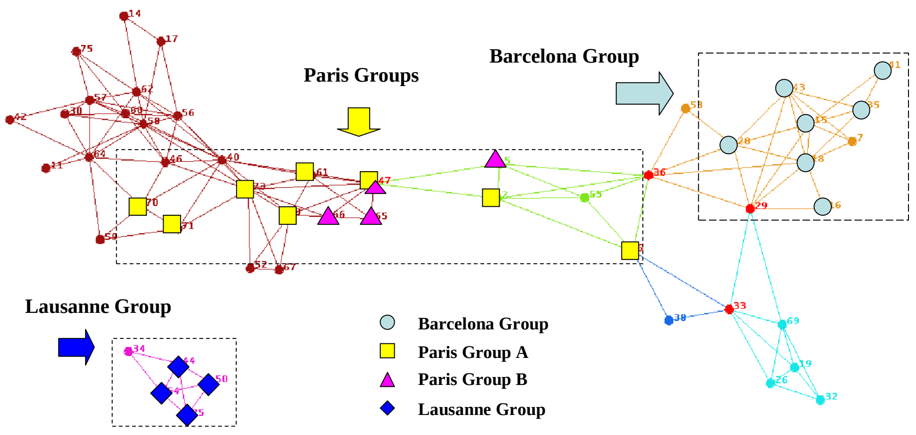


Figure 3.11: 3-clique communities based on contact durations with weight threshold equals 5.6 hours (*Infocom06*).

When I increase k from 3 to 4, the graph splits into 8 communities of size (8, 6, 6, 5, 5, 4, 4, 4). The number of nodes decreases a lot, but we can also see that the cohesiveness of the affiliation communities is quite strong. The Barcelona Group and the Lausanne group are still there, with the numbers change from 7 to 5 and 5 to 4, respectively. The links from node 47 linking two detected communities containing Paris Group members disappear, but we still observe a mixing of five Paris Group A and Group B nodes together to form a community structure.

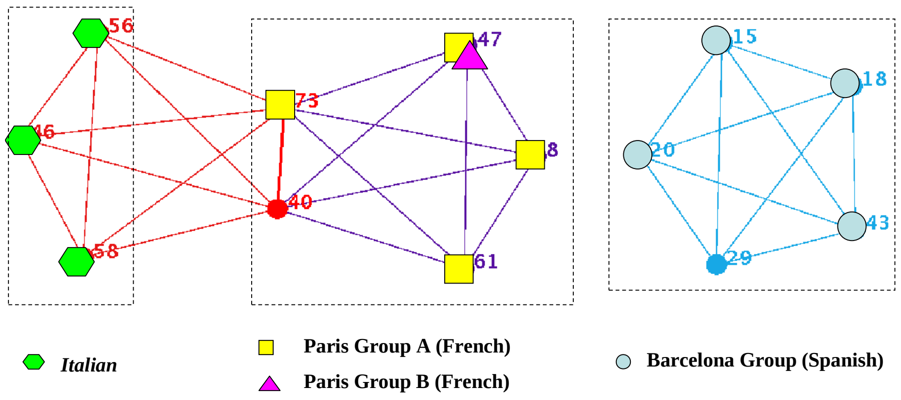


Figure 3.12: 5-clique communities based on contact durations with weight threshold equals 5.6 hours (*Infocom06*).

Figure 3.12 shows the communities when k is equal to 5. There are now only 3 communities of size (5, 5, 5). All small communities with size less than 5 in $k = 4$ case are eliminated. We can observe that the Barcelona Group and a Paris Group are still there. Another group mainly consisting of Italian-speaking people overlaps with the French group. I do not want to claim that the division by K-CLIQUE community algorithm matches real social groups perfectly, but at least it gives us rich information about the underlying human interaction. A preliminary conclusion here is that affiliation or even nationality is strongly linked human contacts, even in the highly mixed environment of a conference.

3.5.4 k-clique Metropolitan Communities

As we can see from Figure 3.5, most pairs have a low number of contacts and low contact duration. I do not expect to discover a rich social structure from this data. However, in this case, we can see how some internal nodes without much social correlation are nevertheless connected together by external Bluetooth devices, by considering all of the 869 nodes detected, including 37 iMotes and 832 external devices.

The experiment lasted 6 days. First I set the threshold to be 3 encounters which is equal to an average of one encounter per 2 days. Around 8% of the total links are taken into consideration. In this case I observed 10 communities sized (18, 10, 8, 6, 6, 5, 4, 3, 3, 3) respectively when $k = 3$, which is shown on the Figure 3.13.

From the same figure we also see that the internal nodes are usually joined together by external nodes. They themselves may not have social correlation at all, but are connected together by these unknown external devices which may belong to colleagues or friends or familiar strangers of the iMote owners. This gives us optimism about the possibility of city-wide PSN data communication. [8]

When $k = 4$, communities shrink to only two small communities of size 4 and 5, respectively. It seems that $k = 4$ is too strong in this case. I tried to increase the number of contacts to be 6, on average one contact per day; in this case only 2.4% of the links are taken into consideration. There are only 6 small communities of size (6, 4, 4, 3, 3, 3), respectively, with only two overlapping with each other at a single node. This again confirms the very sparse social cohesion in the experiment.

The communities detected by the WNA and the K-CLIQUE methods are not necessarily the same, and most probably they will not exactly match because of the overlapping community feature of k-clique. But clearly, they can be useful tools for us to analyse the datasets and extract the important clustering features out of them. I will use these detected communities to test my forwarding algorithms in later chapters.

As I know that there are many community detection algorithms that have been proposed and studied in the literature, the key point is to select the ones which are suitable for our applications. I want to uncover overlapping structures and I also want to reduce manual intervention in choosing the threshold for the data, so I choose K-CLIQUE and WNA. I believe there are some

[8]Although we have observed an expected delay of 1-day in Chapter 2, power-law distribution also implies a lot of short inter-contacts. If we can design communication patterns to make use of these short inter-contacts and avoid or ignore the very long inter-contacts (e.g.night), communication is still possible. I will show that in the next two chapters.

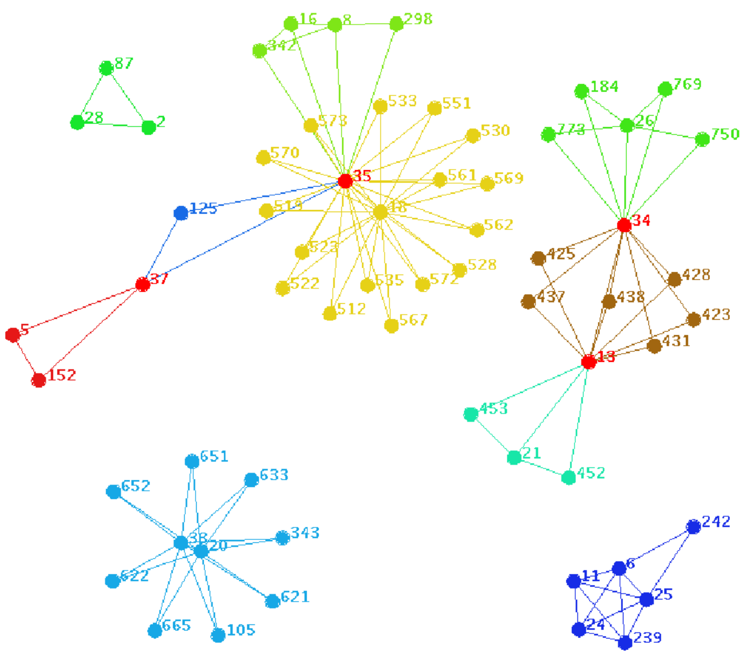


Figure 3.13: Communities based on number of contacts with weight threshold = 3 and k=3 (HK).

other suitable methods, but this work introduces data clustering techniques into human mobility trace studies.

3.6 Distributed Community Detection

The centralised detection methods introduced above are useful for offline data analysis on mobility traces [D$^+$04] [EP06] to explore structure in the data and hence design useful forwarding strategies, security measures, and killer applications. But as self-organizing networks, I would also ask whether the mobile devices can sense and detect their own local communities instead of relying on a centralised server, which leads to the area of distributed community detection. However, to date as far as I am aware, detecting community by distributed approaches has not yet drawn much attention from researchers, especially for mobile applications. Clauset [Cla05] defines a measure of local community structure and an algorithm that infers the hierarchy of communities that encloses a given vertex by exploring the graph one vertex at a time. Although its original design was for static graphs with a known topology instead of dynamic temporal graphs such as PSNs, it provides a motivation for us to examine different centralised community detection algorithms and investigate the possibility of developing a distributed version.

In this section, I will introduce three distributed community detection algorithms, named SIMPLE, k-CLIQUE, and MODULARITY. The difference between these three algorithms are the admission criteria for a node into a local community. SIMPLE uses classic Jaccard similarity [Jac01], k-CLIQUE is based on the clique size, and MODULARITY uses the local modularity measure proposed by Clauset [Cla05]. Except from borrowing local modularity as a fitness measurement of detected communities from the static graph to the dynamic graph in one of the algorithms, all the other parts of the algorithms are my contributions.

In the rest of this section, I will first introduce definitions and terms and then go into the details of these algorithms.

3.6.1 General Framework

Before introducing the basic structure of the algorithms, I will first define some terms which are common to all the three algorithms and also specific to a particular algorithm. As I mentioned above, we can detect different kinds of social communities by specifying the contact duration and number of contacts threshold, but in order to make the presentation easier and in order to compare with other centralised detection methods, which only allow one metric to be specified, I constrain the discussion in this section to the communities detected by contact duration. In the language of graph theory, I refer to a mobile device as a vertex.

The common terminologies for all these algorithms are:
Familiar set: I assume each vertex (mobile device) will keep a map of vertices it has encountered with the corresponding cumulative contact durations. When the cumulative contact duration with a vertex exceeds a certain threshold T_{th}, it is promoted to be included into its *familiar set* F. These two vertices now have an undirected edge between them. A given vertex, v_i, has perfect knowledge of its own familiar set (by definition), denoted F_i. The same vertex also could have gathered incomplete knowledge of other vertices' familiar sets, e.g. a local approximation of the familiar set for vertex v_j would be denoted $\tilde{F}_j$.
Local Community: a vertex's local community, denoted by *C*, contains all the vertices in its *familiar set* (its direct neighbors) and also the vertices that are selected by my following community detection algorithms (the selection criteria of each algorithm to be further elaborated). Because of a lack of temporal synchronisation, each vertex actually in the same community may detect a different local community.

The basic structure of my algorithms is as follows. When a mobile device v_0 first initialises its community detection procedure, the local community C_0 only contains this source vertex. Whenever it encounters another device v_i, they will exchange part of their local knowledge of the network. v_0 then has to decide on the following based on certain acceptance criteria:

1. whether to place the encountered vertex v_i in its *familiar set* F_0 and/or *local community* C_0.
2. whether C_0 should merge with all or part of C_i.

The general framework for the detection algorithms is summarised below:

Algorithm 1: The generic framework of the algorithms (running on vertex v_0)

```
Initialisation: add $v_0$ to $C_0$;
$v_0$ exchanges local network information with $v_i$ (the vertex it encounters);
var $considerMerging \leftarrow false$ ;
while in contact with $v_i$ do
    keep counting the contact histories;
    if $v_i$ can be promoted to $F_0$ then
        put $v_i$ in $C_0$;
        $considerMerging \leftarrow true$;
        break;
if $v_i \notin F_0$ and CommunityAccept($v_i$) then
    put $v_i$ in $C_0$;
    $considerMerging \leftarrow true$
if $considerMerging == true$ then
    MergeCommunities($C_0, C_i$)
```

The three algorithms I introduced here follow this framework. The difference is just the implementation of the functions *CommunityAccept(v_i)* and *MergeCommunities(C_0, C_i)*, and we will look at them in increasing order of complexity from SIMPLE, to K-CLIQUE, to MODULARITY.

3.6.2 SIMPLE

***CommunityAccept** (v_i)* $= true$ *iff*

$$|F_i \cap C_0|/|F_i| > \lambda$$

(where λ is the merging threshold, which I will vary in this chapter to see the differences of the final communities detected). See Figure 3.14.

MergeCommunities(C_0, C_i): if v_i is added to C_0, we will consider merging the two local communities *iff*

$$|C_0 \cap C_i| > \gamma |C_0 \cup C_i|$$

See Figure 3.15.

As its name implies, this is a really *simple* approach. In essence, we admit a vertex v_i into our local community C_0 (even if it is not in the familiar set of v_0) if most of its connection are to vertices that are in our local community already. And we merge the two local communities if the main parts of them already overlap (See Figure. 3.15). Both functions *CommunityAccept(v_i)* and *MergeCommunities(C_0, C_i)* preserve the intuition that there should be more intra-community connections than inter-community ones.

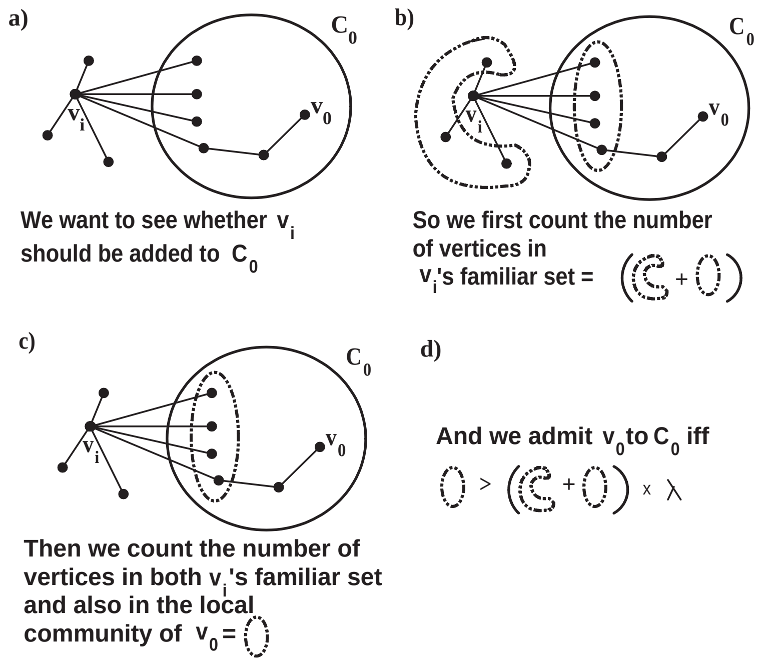

a)
b)
c)
d)
C0
vi
v0
We want to see whether vi should be added to C0
So we first count the number of vertices in vi's familiar set =
Then we count the number of vertices in both vi's familiar set and also in the local community of v0 =
And we admit v0 to C0 iff
x λ

Figure 3.14: Admission criteria for SIMPLE

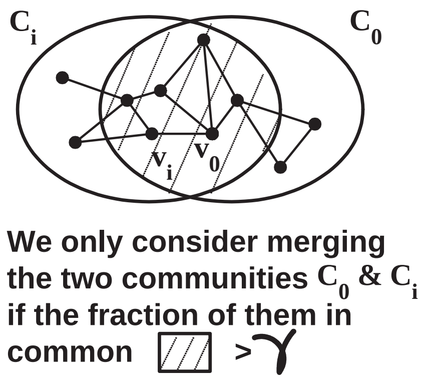

Ci
C0
vi
v0
We only consider merging the two communities C0 & Ci if the fraction of them in common > γ

Figure 3.15: Communities merging criteria for SIMPLE

3.6.3 k-CLIQUE

CommunityAccept **(**v_i**)** $= true$ *iff*

$$|F_i \cap C_0| \geq k - 1.$$

i.e. the *familiar set*, F_i contains at least $k-1$ members of the local community, C_0, (See Figure 3.16).

MergeCommunities(C_0***,*** C_i***)***: if v_i is added to C_0, we consider each vertex v_j inside C_i (the local community of v_i), and if *CommunityAccept(*v_j*)* (its *familiar set*, $\tilde{F}_j$ contains at least $k-1$ members of C_0), v_j is also added to the local community C_0, i.e. the admission criterion for

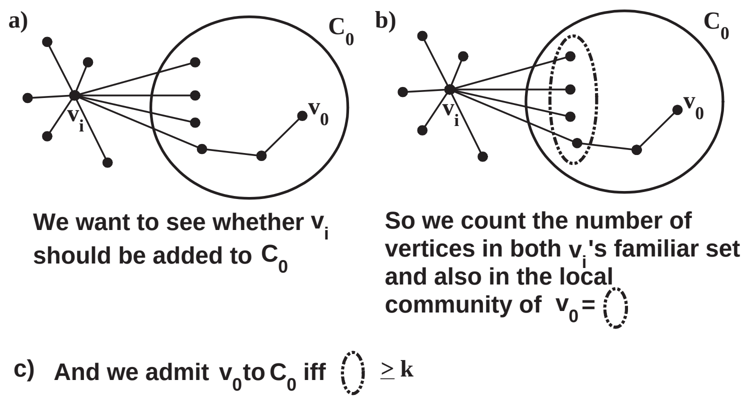

a)
C_0
v_i
v_0
We want to see whether v_i should be added to C_0
b)
C_0
v_i
v_0
So we count the number of vertices in both v_i's familiar set and also in the local community of v_0 =
c) And we admit v_0 to C_0 iff ≥ k

Figure 3.16: Admission criteria for k-CLIQUE

each v_j is

$$|\tilde{F}_j \cap C_0| \geq k - 1$$

This approach is based upon the concept of k-clique communities [PDFV05], where each community is a union of *k-cliques* (smaller complete, fully connected, subgraphs of k nodes) that can be reached from each other through a series of *adjacent k-cliques*, where two k-cliques are said to be adjacent if they share $k - 1$ nodes.

3.6.4 MODULARITY

Clauset introduced the concept of Local Modularity [Cla05] for distributed community detection. The following are the relevant definitions (Figure 3.17):

Boundary Set: For a given vertex v_0 and its local community C_0, the associated boundary set B_0 is defined as the subset of vertices in C_0, whose members have edges connecting to one or more vertices outside C_0, i.e.

$$B_0 = \{v_i \mid (v_i \in C_0)\ and\ ((F_i \setminus C_0) \neq \emptyset)\}$$

Local Modularity: The local modularity R for a given C_0 with B_0 is defined as

$$R_0 = \frac{I}{|T|}$$

where T is the set of edges with one or more endpoints in B_0, while I is the number of those edges with both of their endpoints in C_0. If $B_0 = \emptyset$, R_0 is defined to have value of 1.

Each vertex can use Local Modularity as a measure of the sharpness of its local community boundary, and the measure is independent of the size of the enclosed community.

CommunityAccept (*v_i*) $= true$ *iff* $(F_i \neq \emptyset)$ and either
a) $(F_i \subseteq C_0\ and\ B_0 \neq \emptyset)$ or

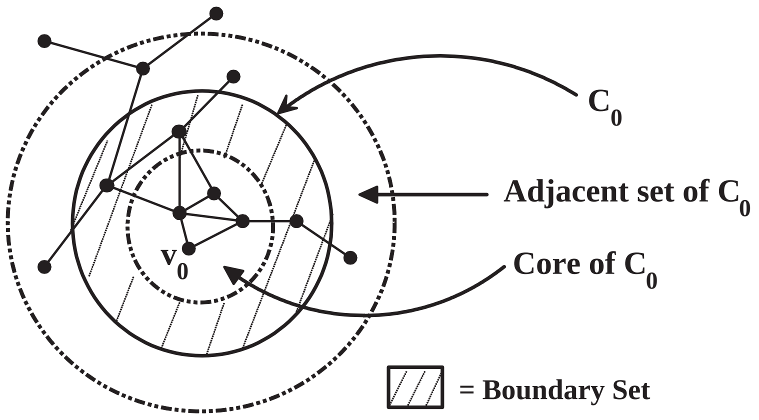


Figure 3.17: The higher the local modularity of a community, the fewer the number of edges connecting it to its Adjacent Set. A community has a Local Modularity value of 1 when it has an empty Boundary Set.

b) $\Delta R_0^i > 0$ (the difference between the local modularity measure before and after adding v_i to C_0 is +*ve*).

MergeCommunities(C_0***,*** C_i***)***: the algorithm only considers adding the vertices in the set K :

$$\{v_k \mid \text{there exist } j \text{ s.t. } v_j \in C_0 \cap C_i \text{ and } v_k \in \tilde{F}_j \text{ and } v_k \in C_i \setminus C_0\}$$

For each $v_k \in K$, it evaluates whether $\tilde{F}_k \subseteq C_0$. If this condition is satisfied, the corresponding v_k is added to C_0. The rest of the v_k are then considered in descending order of ΔR_0^k. Vertices with a negative or zero contribution to ΔR will not be added to C_0, and the values of ΔR are re-evaluated after each addition to C_0.

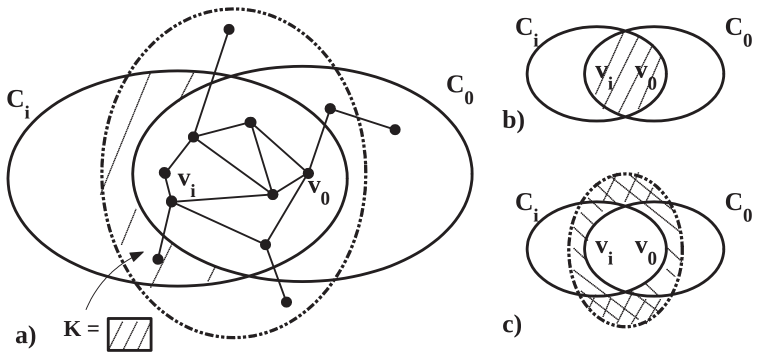


Figure 3.18: Explanation of set K in MODULARITY

Figure 3.18 illustrates how and why set K is chosen. The shaded area in Figure 3.18(a) shows the vertices in the set K. When considering merging parts of two communities together, one first considers locating all the vertices that are common to the local communities of both vertices ($C_0 \cap C_i$, shaded area in Figure 3.18(b)). Then we consider those vertices that are adjacent to the set of vertices in $C_0 \cap C_i$ (shaded area in Figure 3.18(c)), which are more closely connected to the common vertices of the two local communities and hence suitable for merging into C_0. But since a portion of them are already in C_0, and a portion of them are in neither of the local communities, we only consider the set of vertices in K.

Both functions *CommunityAccept(υ_i)* and *MergeCommunities(C_0, C_i)* make sure that the ratio of intra-community connections to inter-community ones would always increase for each addition of a community member.

Clearly, the SIMPLE algorithm requires less storage and less computation. The k-CLIQUE algorithm is in the middle and MODULARITY is the most demanding one - because of the need to re-evaluate ΔR in each iteration, hence in *MergeCommunities(C_0, C_i)* only part of the community (K) is considered to be merged, as a resource/performance tradeoff.

3.7 Evaluation of Distributed Detection

In this section, I evaluate the communities detected by the distributed methods against the centralised methods. In order to do the comparison, I need to first develop similarity measurements.

3.7.1 Similarity Measures

Newman [New04a] introduce a metric called *fraction of vertices correctly identified* to evaluate the communities detected against pre-known communities. According to the definition, the largest set of vertices that are grouped together by the algorithm in each of the known communities is found first. If the algorithm puts two or more of these known communities in the same set, then all vertices in those sets are considered incorrectly classified. Otherwise, they are considered correctly classified. All other vertices not in the largest sets are considered incorrectly classified.

Another measurement metric is used in [DDDGA05] by Danon *et al.*, which is called *normalised mutual information* measure. It is based on defining a confusion matrix **N**, where the rows correspond to the "real" communties, and the columns correspond to the "found" communities. Each element of **N**, N_{ij} is the number of nodes in the real community *i* that appear in the found community *j*. The *normalised mutual information* is then defined as following:

$$NMI(A,B) = \frac{-2\sum_{i=1}^{c_A}\sum_{j=1}^{c_B} N_{ij} log(\frac{N_{ij}N}{N_{i.}N_{.j}})}{\sum_{i=1}^{c_A} N_{i.} log(\frac{N_{i.}}{N}) + \sum_{j=1}^{c_B} N_{.j} log(\frac{N_{.j}}{N})} \tag{3.7}$$

where c_A is the number of real communities, c_B is the number of found communities, $N_{i.}$ is the sum over row *i* of matrix N_{ij} and $N_{.j}$ is the sum over column *j*.

I can adapt these two measurements to evaluate my distributed community detection algorithm against its corresponding centralised method. However before moving forward, I need to consider the fairness problem. Since, for distributed community detection, each node will detect the local communities to which it belongs, if a system has N nodes, there would be at least N communities detected (i.e. $c_B \geq N$). Denote the number of real communities as c_A as above. If $N \gg c_A$, the evaluation of c_A against c_B would be unfair, especially if over weighted by the big community. Also, considering that the network is a temporal graph and some nodes are more popular than others, nodes belonging to the real community may not have the same local view of the communities detected. Therefore we need to consider a modification to address this problem. My approach is to choose the biggest detected community, move it to the *core community* list and then discard the communities detected by all the nodes included in it, and then repeat this for the remaining biggest one on the list and continue until no more communities are left. I then evaluate c_A against the *core community* list. The biggest communities are not necessarily the best communities detected: they may contain a lot of redundancy so the selection of the biggest communities does not necessarily favor my algorithms. This shrinking process will remove the smaller groups of the overlapping communities, which may also penalise my results.

Newman's method is a little bit harsh; as he mentioned in his paper, there are cases in which one might consider some of the vertices to have been identified correctly, and this method would not. Also considering that for all three datasets, there are many more single-node communities than bigger communities, this will make the NMI measure tend towards 1. Hence here I consider another modified similarity measurement. Here I introduce similarity by using the classic Jaccard index [Jac01] which was proposed by Jaccard over one hundred years ago to evaluate the similarity of two communities.

$$\sigma_{Jaccard} = \frac{|\Gamma_i \bigcap \Gamma_j|}{|\Gamma_i \bigcup \Gamma_j|} \tag{3.8}$$

where Γ_i is the members of community *i* and $|\Gamma_i|$ is the cardinality of the set Γ_i, that is equal to the number of members in community *i*. In this chapter, I will compare the *core communities* detected by distributed methods with the communities detected by centralised algorithms using this similarity measurement.

3.7.2 Results of Detection

To evaluate the community detection algorithms, I replay the mobility traces of the three experiments and emulate the gossiping of community information on each encounter. Here I only evaluate the communities detected after the whole traces, which lasted 9 months for *Reality*, 3 months for *UCSD* and 11 days for *Cambridge*. As a first step I do not evaluate the time needs for the communities to be well developed at the middle of the emulation.

Figure 3.19 shows the similarity between the communities detected by the distributed SIMPLE method and the k-CLIQUE against the communities detected by the centralised k-CLIQUE algorithm with a threshold of 389k seconds for the *Reality* dataset, 78k seconds for the *UCSD*, and 36k seconds for *Cambridge*. These threshold values for the centralised methods were selected

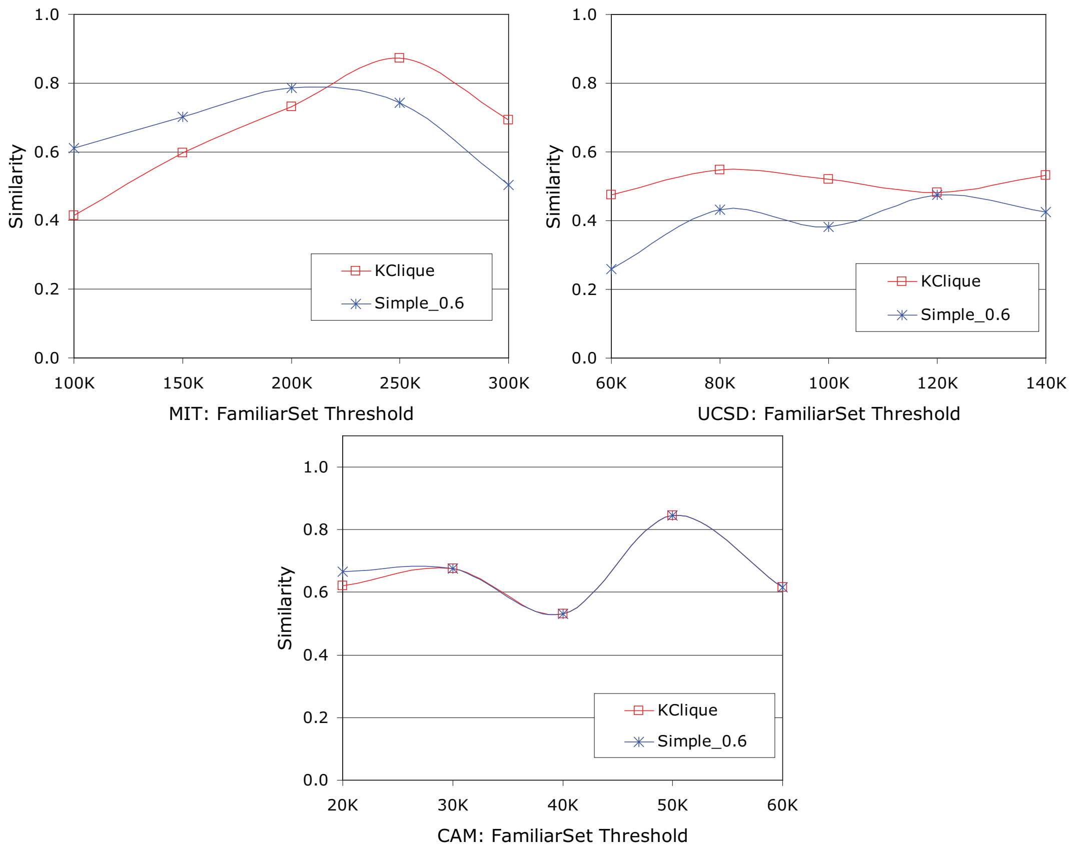


Figure 3.19: Impact of Familiar Set threshold, k-CLIQUE and SIMPLE

following many trials and also by studying the nature of the group of experimental objects. Some of them are found to agree with the real experimental groups, such as for the *Cambridge* data, the two groups detected correspond to the two main participant groups. Figure 3.19 also shows the different similarity values with different familiar set thresholds. We can see that the k-CLIQUE method shows better results most of time than the SIMPLE method. With a suitable threshold, the distributed algorithms for both SIMPLE and k-CLIQUE can reach around 80% of the performance of the centralised algorithm. For the SIMPLE case, I use a merging threshold, λ, of 0.6. I also find out that varying the merging threshold from 0.5 to 0.9 makes little difference; whereas the Familiar Set threshold changes the similarity values quite significantly. Figure 3.20 shows an example for the *Reality* data using the SIMPLE approach.

Since we know the network is highly intermittently connected, the local community information for each node within the same community may not be synchronised. I want to know how different these local community views are. From the *Core Communities*, we can compare the local community detected by each member in each *Core Community* with its *Core Community*, calculate the similarity values and then plot the distributions of all these similarity values. Figure 3.21 shows these distributions for the three datasets (they are *Reality*, *UCSD* and *Cambridge* respectively) using distributed k-CLIQUE. We can see that for both *Reality* and *Cambridge*, the local community views are quite similar to the selected largest *Core Community*. This would be probably because of the relatively smaller dataset size and higher connectivity. And these

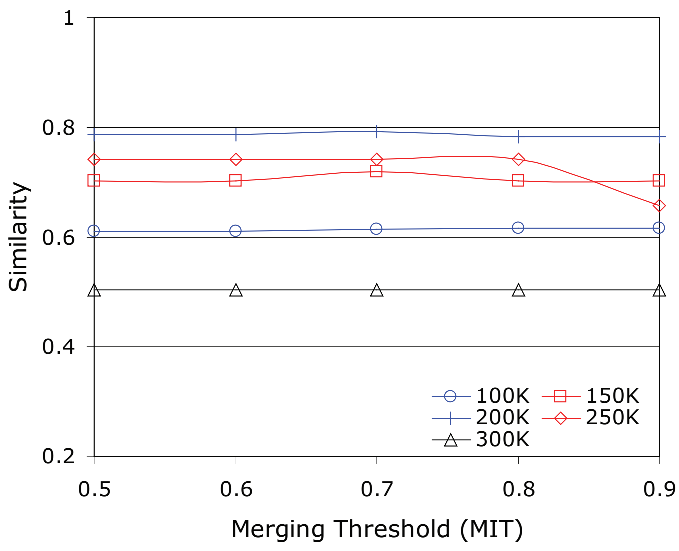


Figure 3.20: Impact of merging threshold

two groups of nodes also agree with the two main groups of students participating in the experiment.The *UCSD* data relies on users connecting to centralised access points instead of peer contacts and hence is more sparse. In general, the local community views have bigger variation when using SIMPLE. Figure 3.22 gives an example of the *Reality* case. Similar variations are also observed in the other two datasets. And I also find out that there is no impact of the merging threshold, λ (from 0.5 to 0.9) on the distribution if using the same Familiar Set threshold.

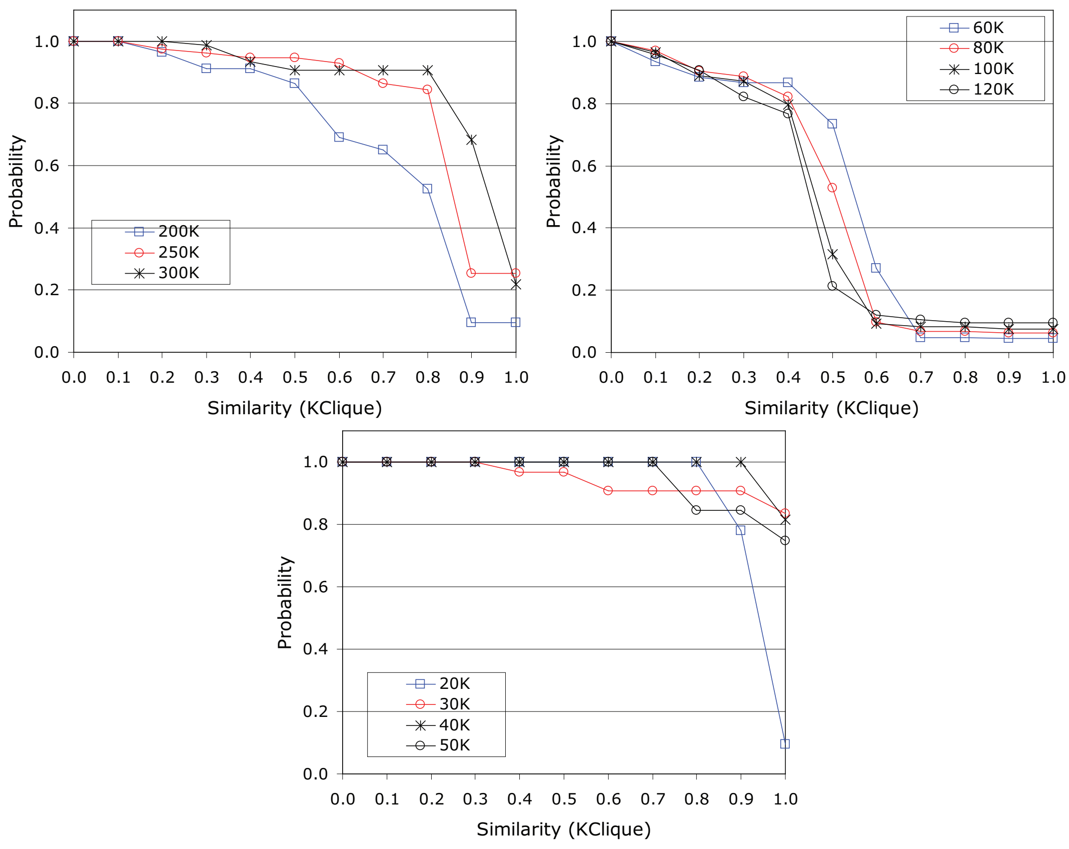


Figure 3.21: Distribution of local community views

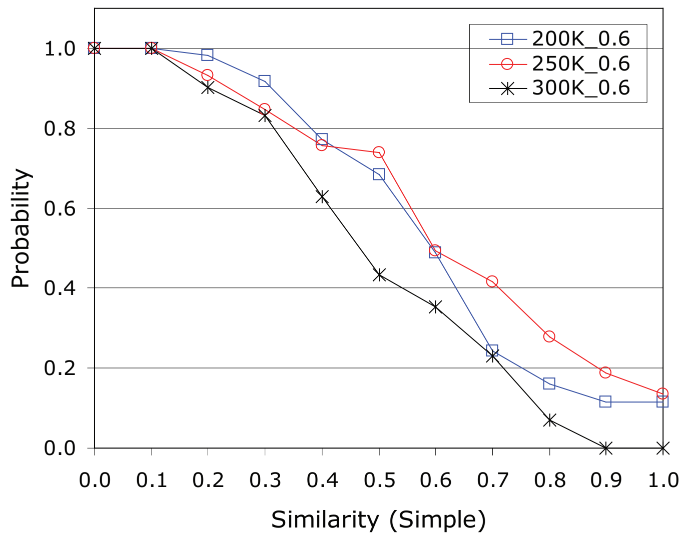


Figure 3.22: Distribution of local community views SIMPLE

Figure 3.23 also shows the comparisons of MODULARITY and SIMPLE with centralised Newman WNA [J.N04]. Since the MODULARITY and Newman methods both used modularity, it is fairer to compare them than comparing with the centralised k-CLIQUE method. We can see that in the *Reality* case, MODULARITY has better performance than SIMPLE with the same threshold, and it has a best performance of around 80% similarity. The SIMPLE approach also generally has good performance with a 75% similarity for a suitable threshold setting. For *UCSD*, the similarity values are in general low for both algorithms and this is also true for the k-CLIQUE algorithm. For the *Cambridge* data, at high threshold values, MODULARITY behaves better than SIMPLE and the other way round at low threshold values, but they both reach a maximum point above 80%.

I conclude this section with Table 3.2, which summarises the highest similarity values calculated by each distributed algorithm. For SIMPLE, I show both its comparison with the centralised k-CLIQUE (first) and the centralised Newman method (second). We can see that generally MODULARITY and k-CLIQUE have slightly better accuracy (i.e. more similar to the centralised methods) than their SIMPLE counterpart. That is to be expected since they require more information and calculation, especially as the computation complexity of MODULARITY is $O(n^4)$ in the worst case, where n is the size of the network explored so far. However, since a factor of n^2 is contributed by the evaluation of each ΔR, which in reality is likely to be bounded by $O(k^2)$ where k is the average degree of a vertex in the graph, the worst case performance is thus $O(n^2k^2)$. Considering its computational and storage requirements, the performance of SIMPLE is quite acceptable, so I would suggest SIMPLE, with $O(n)$, for the mobile devices with strong constraints on storage and computational complexity. If the mobile devices can afford the storage for a local copy of the Familiar Set of their community members, k-CLIQUE would be a good choice for its reasonably good similarity values and also quite low computational complexity, $O(n^2)$ in the worst case. MODULARITY requires the most computational power but it does not have significant better performance in these cases. This may be biased by the

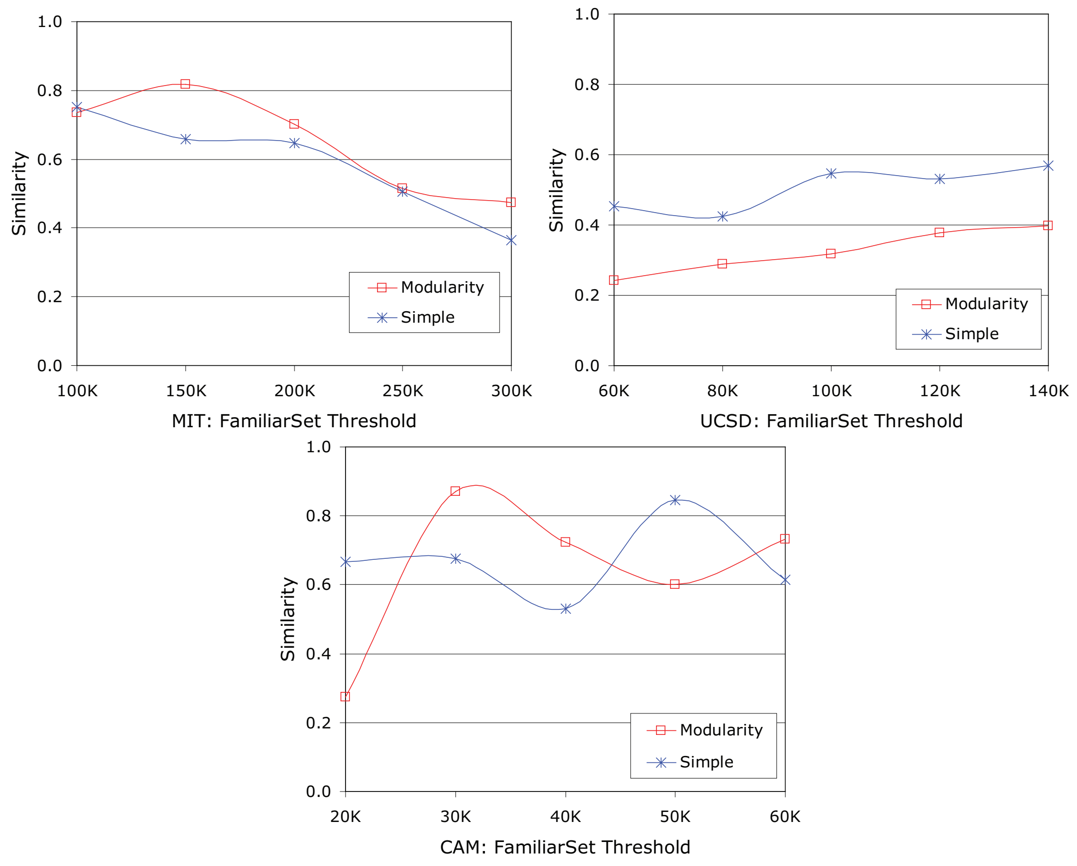


Figure 3.23: Impact of Familiar Set threshold, MODULARITY and SIMPLE

limitations of the experimental datasets, but I will not strongly recommend it at this moment.

Experimental dataset	SIMPLE	k-CLIQUE	MODULARITY
Reality	0.79/0.76	0.87	0.82
UCSD	0.47/0.56	0.55	0.40
Cambridge	0.85/0.85	0.85	0.87

Table 3.2: Summary of best performance of the algorithms

3.7.3 Limitations

There are several limitations to my study in this chapter, and I want to point them out here:

- As a first study, I only evaluate the communities detected after the replaying of the whole traces, but did not evaluate the communities at different stages of the emulation. Evolution of the communities detected at different times could also be an interesting study topic.

- The Familiar Set threshold values I used in the emulations are trace-dependent and were chosen based on the whole duration of a trace. In a real application, we may want to specify them in more general terms such as number of hours or number of times per day, per week or per month. Here I just want to compare the performance of the distributed algorithms with the centralised ones so I simply specify them in relation to the whole experimental durations.

- I did not evaluate the detection of different categories of relationship in this chapter. In real applications, however, the mobile devices should be able to detect the different categories of relationship in Figure 3.4 by specifying the Familiar Set thresholds for contact durations and number of contacts.
- In the current version of the algorithms, I need to specify a static Familiar Set threshold, but maybe in future versions more dynamic methods such as weight-averaging, which dynamically choose the threshold by considering the average value of the weights over the edges connecting all the neighbours, could be used to reduce manual configuration.
- I did not consider *aging* of the contacts at this moment, but we need to look into it in the future. Some previous contacts may become irrelevant after some time, but which take up storage and cause false-positive impact for detection, so good *aging* mechanisms about contacts should be considered.

3.8 Conclusion and Future Work

I have applied Newman's WNA and K-CLIQUE community detection to several PSNs and found the communities detected match well with real social communities. These two offline analysis tools would be very useful for us to analyse and extract human interaction patterns from all experimental datasets. There are a lot of community detection algorithms in the area of complex networks; one key point is to pick the correct ones for a particular application. I believe that there are other algorithms which can also serve the same purpose. But my most important contribution here is opening a new aspect of mobility trace analysis that the research community can follow and further improve and a lot of different research can be done based on it. For example, we can use these detected communities to test a series of social-aware forwarding algorithms, data sharing overlay design, and security policies in the future. In the later chapters, I will evaluate the impact of the detected communities on PSN forwarding efficiency compared with "oblivious" flooding or using randomly generated groups with the same size.

I also proposed three distributed community detection algorithms with different levels of computational complexity and resource requirements. I evaluated them on three human mobility experimental datasets with Bluetooth peer-to-peer and WiFi client-access point logging traces. I discovered that the communities detected by the distributed algorithms can satisfactorily approximate the centralised algorithms which require the whole network topology. I also compared the performance of these three algorithms and proposed a scenario in which each could be used. These distributed algorithms are not part of offline data analysis but they provide the possibility for each mobile device to identify its local community and hence can be used for online applications. To the best of my knowledge, this is the first work that uses mobile devices to infer human communities.

In future, I would like to evaluate my algorithms on more mobility traces, such as the WiFi traces from the Crawdad project [HKA04], and also some forthcoming iMote experiments to make a more conclusive statement about the accuracy and application scenarios of these algorithms. And I would also like to develop my studies further with regard to the limitations I listed in section 3.7.3.

Chapter 4

How Small Labels Create Big Improvements

Following up the mathematical and empirical analysis of “oblivious” forwarding algorithms in Chapter 2, I want to start looking at using social context to improve forwarding efficiency in this chapter. Society naturally divides into communities according to needs for cooperation or selection. In sociology, the idea of *correlated interaction* is that an organism of a given type is be more likely to interact with another organism of a same type than with a randomly chosen member of the population [Oka05]. If the correlated interaction concept applies, then our intuition is that using this community information to influence forwarding paths may be advantageous. In this chapter, I study the impact of affiliation labels on PSN forwarding. To the best of my knowledge, this is the first empirical work in this area. This is a joint work with my supervisor Prof. Jon Crowcroft. Dr. Meng-How Lim, Dr. James Scott, Dr. Augustin Chaintreau, Richard Gass, Dr. Christophe Diot, and Dr Eiko Yoneki were also involved in the organizing of this experiment.

4.1 Introduction

In Section 2.6, I have shown empirically that traditional naive multiple-copy-multiple-hop (MCP) flooding schemes work well in dense environments such as academic conferences, and provide fair performance in sparser settings in terms of delivery ratio and delay. However, in terms of delivery cost, the naive approach is far from satisfactory, as it creates a lot of unwanted traffic as a side-effect of the delivery scheme, and the overhead rapidly becomes unacceptable in a mobile network characterized by resource scarcity, vulnerability, and contention.

In the research community, it has been a widely held belief that identifying community information about recipients can help in selecting suitable forwarders, and reduce the delivery cost compared to “oblivious” flooding. This is a reasonable intuition, since people in the same community are likely to meet regularly, and hence be appropriate forwarders for messages destined for other members of their community. However, to date as far as I am aware, there has been no experimental evaluation of this belief, and no one knows whether it is valid or not.

I conducted a human mobility experiment during IEEE Infocom 2006, with the participants labeled according to their academic affiliations. After collecting 4 days of data during the conference period, I replayed traces using an emulator, and discovered that a small label indicating affiliation can indeed effectively reduce the delivery cost, without trading off much against delivery ratio. The intuition that simply identifying community can improve message delivery turns out to be true even during a conference, where the people from different sub-communities tend to mix together.

Inspired by the inter-contact time analysis, I also proposed an early model for temporal graphs based on community and temporal attachment.

The rest of this chapter is arranged as follows. I first introduce related work in Section 4.2; then followed by the "labeling" strategy in Section 4.3. In Section 4.4, I analyse the inter-contact time within communities and between communities. I introduce the evaluation methodology and present the results in Section 4.5 and Section 4.6 respectively. I also introduce a simple model for temporal graphs with community topologies and power-law temporal attachment in Section 4.7. Finally I give a short conclusion of the chapter.

4.2 Related Work

Forwarding strategies under intermittently connected mobile ad hoc networks have been explored by a number of research groups. I have presented an analytical foundation on the impact of human mobility on the design of opportunistic forwarding algorithms based on six real human mobility traces from four different research groups in Chapter 2 (also see [CHC$^+$06]). Lindgren *et al.* considered the community concepts for controlled flooding [LDS04]. They have the assumption that nodes mainly remain inside their community and sometimes visit others. To route a message to a destination, a node may transfer that message to a node that belongs to the same community as the destination. Their work provides a good theoretical hypothesis for community-based routing, but there has not been any empirical evaluation. Musolesi *et al.* proposed a community-based mobility model for mobile ad hoc research [MM06]. In [HCS$^+$05], we took a similar measurement of human mobility in a conference environment but community issues have not been touched. In this chapter, I am looking at the problem using an experimental approach. Empirical results could be helpful for both modeling and theoretical work of other research groups.

4.3 Labeling Before Throwing out

The forwarding scheme I used here is called the LABEL strategy. We imagine that each node has a label telling others about its affiliation/group, just like the name badge in a conference. The strategy chosen is exclusively to forward messages to the destination, or to next-hop nodes belonging to the same group (same label) as the destination. The assumption I make here is that people with same affiliation tend to meet more often than people outside the affiliation, and hence can be good forwarders to relay messages to the other members in the same affiliation/with the same label. This is similar to the *correlated interaction* concept in sociology [Oka05]. This strategy requires little information about each individual and is believed to be easy to implement in real life, by just “tapping” (imagine when we use a small pen to hit on the touch screen a PDA and write on it) mobile device and writing down the affiliation of the owner, which is what we are usually required to input when we have a new PDA.

Here, I do not require a node to know all the other nodes with the same label. Two encounter nodes only need to know whether they have the same label. This approach is much more scalable since each node only need to store its own label but not information about other nodes with the same label. This is doable if there is a pre-agreed labeling scheme (e.g. affiliation[1]), or I call it *explicit community*, which can be explicitly named. For some communities, which cannot be explicitly named (*implicity community* [2]), each device can detect its own community by using distributed community detection algorithms I have introduce in Chapter 3. In this chapter, I will limit to study *a priori* affiliation groups, and leave all the detail discussion about communities in the next chapter.

4.4 Analysis of Inter-contact Times

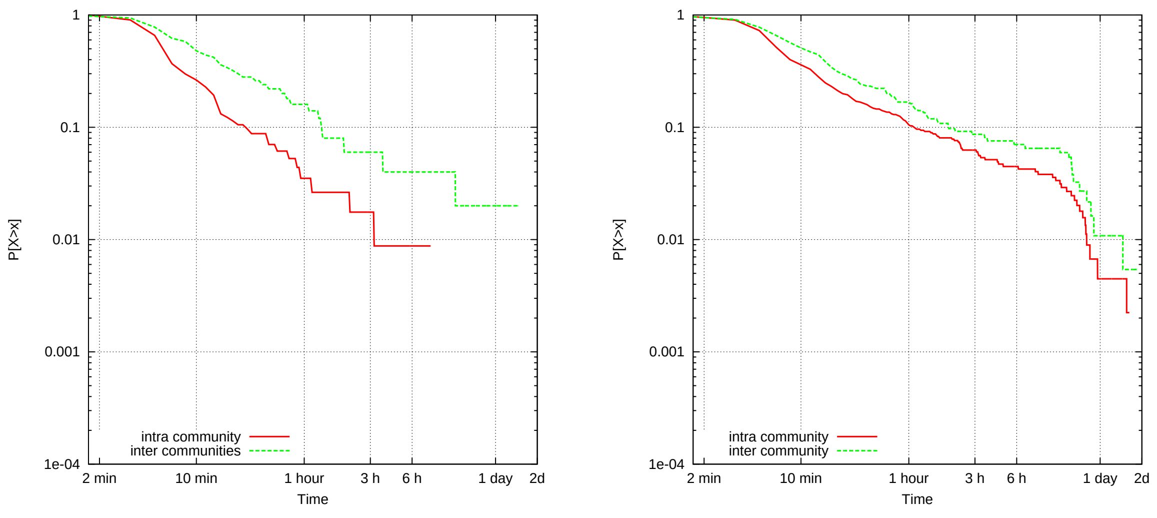
Before moving into the detailed performance evaluation of LABEL, I want to first verify the correlated interaction from the dataset with *a priori* group information. Inter-contact time distribution is a good indication for this relationship. For a given pair of nodes A and B, the time-line can be divided into two regions, contact times and inter-contact times. The contact times are when A and B are in range of one another, and could therefore have sent data if they had wished to. Inter-contact time is the time elapsed between two successive contact periods for a given pair of devices (see Section 2.4), hence the inter-contact time distribution simply indicates the frequency of interaction. In last chapter (also [HCS$^+$05]), I have shown that inter-contact time follows a power-law distribution, the bigger that value of the power coefficient, the more frequently the node pairs interact. In this work, I extend this to look at the inter-contact distribution for all the nodes inside a group and also the inter-contact distribution between two

[1]All the Intel employees in the world share the same Intel label.

[2]For example, Bob usually hang out together with some friends from his high school, some colleagues from his company, and also his neighbour, but there is not an explicit label to describe this group.

groups. I believe the power-law coefficient of these inter-group inter-contact time distributions, if they are following a power-law, indicates the closeness of two groups.

Figure 4.1(a) shows a typical inter-contact time distribution for a pair of nodes in one community, and a pair of nodes from different communities. We can see that the intra-community pair has a higher power-law coefficient than the inter-community pair; that is, node pairs in the same community tend to meet more often. Figure 4.1(b) also shows the aggregated inter-contact time distribution for a node with all the other nodes in its community, as well as for all other nodes in another community; we can also observe a variation in the power-law coefficient. To avoid the bias caused by a single node, I also calculate the aggregated inter-contact time distribution for all the nodes within the same group and also the same distribution with all nodes outside the group; the results are shown in Figure 4.2. We can see a significantly steeper slope for the intra-community aggregated inter-contact distribution. This provides empirical evidence that people from the same organisation tend to meet more often than people from different organisations. This provides good hints to identify forwarders for message delivery.



(a) inter-contact (b) aggregated inter-contact

Figure 4.1: Comparison of inter-contact and aggregated inter-contact time for intra and inter community nodes

Here I also want to introduce the concept of friendship communities [Dun98]. The University of Cambridge's Computer Laboratory researchers may be a friendly community towards Intel Research Cambridge staff, since these two groups of people use the same building[3] and have a number of collaborations. Hence people from one group may be good forwarders for people in the corresponding friendship group. In the experimental data, there are two groups from Paris, and I want to look at whether they have a closer relationship when compared to other groups, based on the inter-contact time distribution. As shown in Figure 4.3, we can see that within the same group, the power-law coefficient is the largest, and next largest with the nodes in a friend-

[3]We do not share a building anymore!!

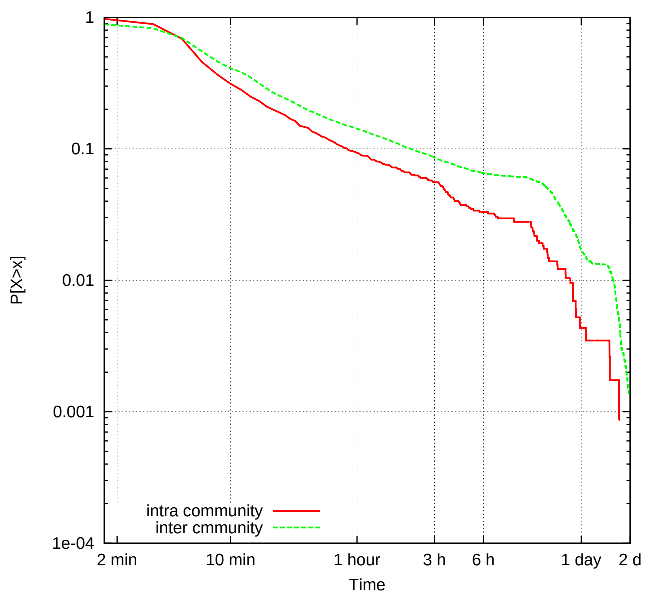


Figure 4.2: The aggregated inter-contact time distributions for all nodes inside a same group and also with all other nodes outside the group.

ship group, and lowest for an arbitrary group. Although the difference is not very significant, we can still observe it. Later, I will also look at how helpful this friendship community can be when used explicitly to forward messages.

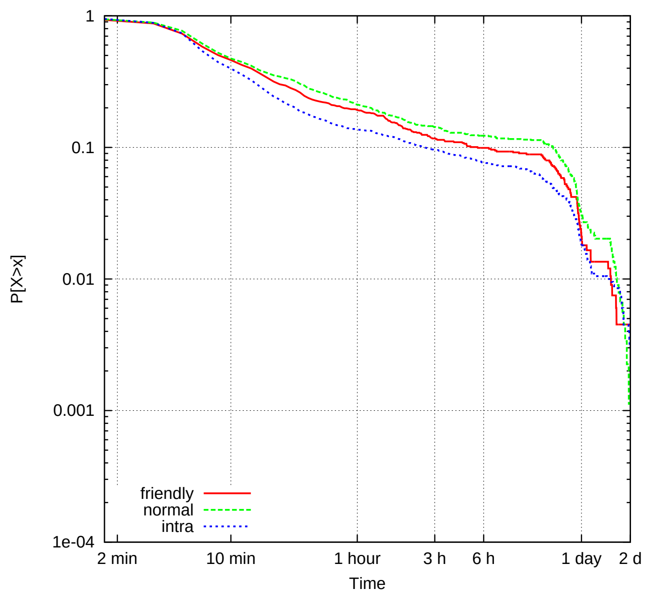


Figure 4.3: The aggregated inter-contact time distributions for all nodes inside a same group, with a friendship group and a normal group.

4.5 Evaluation Methodology

4.5.1 HaggleSim Emulator

In order to evaluate different forwarding algorithms, I developed an emulator called *HaggleSim*, which can replay the mobility traces collected and emulate different forwarding strategies on every contact event. This emulator is driven by contact events. The original trace files are divided into discrete sequential contact events, and fed into the emulator as inputs. The event granularity depends on our choice of balance between replay speed, and the degree of accuracy desired. In all the simulations in this work, I divided the traces into discrete contact events with granularity of 100 seconds. I analyse the successful delivery rate, the delivery cost, the delay distribution, the hop count distribution for all successful deliveries, and the popularity of a node as a relay based on the log files produced by the emulator.

Table 4.1 shows a snapshot of my emulation source file. My simulator reads the file line by line, treating each line as a discrete encounter event, and makes a forwarding decision on this encounter based on the forwarding algorithm under study. As we can see from the source file, some events happen at the same time stamp, and should be treated as simultaneous. Just reading the file line by line artificially imposes an order on events, so instead I keep contacts in a buffer while reading multiple lines ahead: if the next event happens at the same time, and there would be an exchange of messages between the nodes referenced in the line due to the forwarding strategy, I will re-read the contacts in the buffer, and apply the same forwarding strategy to the newly exchanged messages. This removes the artificial ordering.

Node	**Node**	**Time Stamp**
6	24	14991
25	11	14991
25	240	14991
24	240	14991
6	240	14991
35	511	14991
24	11	14991
6	24	15001
25	11	15001
25	240	15001

Table 4.1: Snapshot of Simulation Source File

4.5.2 Simulation Parameters

There are three parameters I used in my simulation to achieve controlled flooding in MCP strategy. [4]

- *Number of copies (m)*: The maximum number of duplicates of each message created at each node.
- *Number of hops (Hop-TTL)*: The maximum number of hops, counted from the source, that a message copy can travel before reaching the destination; this is similar to TTL value in the Internet.
- *Time TTL*: The maximum time a message can stay in the system after its creation. This is to prevent expired messages from further circulation.

4.5.3 Performance Metrics

For all the simulations I have conducted for this work, I have measured the following metrics:

- *Delivery ratio*: The proportion of messages that have been delivered out of the total number of messages created.
- *Half-life delivery time TTL*: This is the time TTL value that would allow half of the messages created to be delivered; in other words it is equivalent to the delay time that half of the created messages experienced. It measures how fast and efficient a forwarding strategy is for message delivery.
- *Hop-count-distribution for deliveries*: The distribution of the number of hops needed for all the deliveries. This metric gives some idea of how a forwarding strategy picks forwarders. In the LABEL strategy, it reveals the social distance between sources and destinations.
- *Delivery cost*: For cost, I measure the total number of medium accesses; that is the total number of messages (includes duplicates) transmitted across the air. To normalise this, I divide it by the total number of unique messages created.[5]

[4]Same as the evaluation of MCP in Section 2.6

[5]Packet loss in the wireless environment will add more cost, but I will not look at it in this work, as I have explained in Chapter 2.

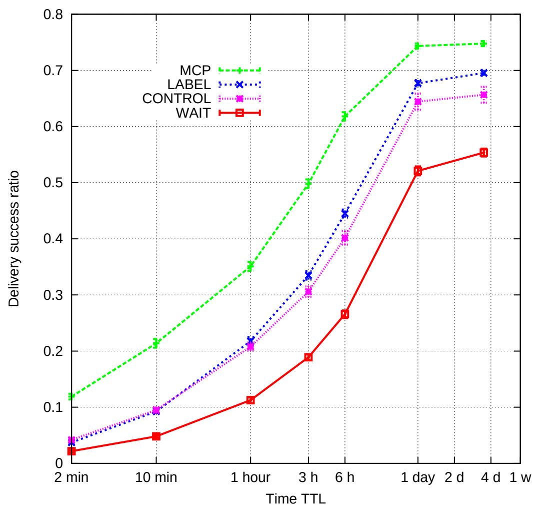
4.5.4 Simulation Scenario

In order to study exclusively the effect of community on forwarding, I created the following scenario: all the seventy-seven nodes [6] create a total of 1000 messages, destined only to the thirty-four nodes belonging to the four groups; the message creation times are uniformly distributed throughout the experimental duration.

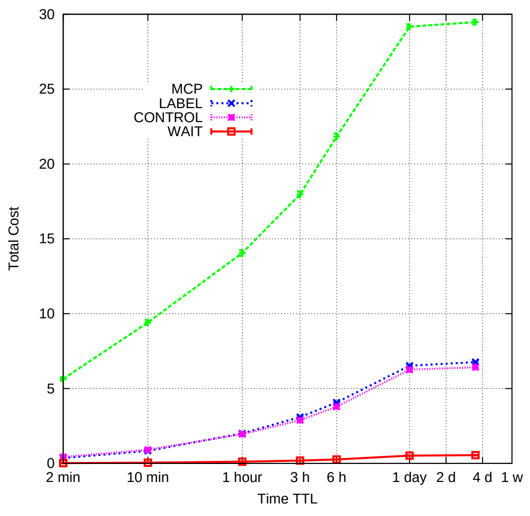
In order to compare the performance of the labeling strategy with a naive strategy, I run an emulation of the MCP strategy. To ensure that the performance improvement is not due to arbitrary limited number of forwarders, for every round of simulation, I created four random groups of same group sizes as the original groups but with nodes randomly selected from all the seventy-seven nodes. I refer to the labeling strategy and the control experiment as *LABEL* and *CONTROL* respectively, in my analysis. To achieve statistical fairness, I run the emulation 20 times with different traffic patterns.

4.6 Results and Analysis

In this section, I compare the performance of four strategies, MCP, LABEL, CONTROL, and *WAIT* (*wait-and-forward*); this entails waiting until the source of a message has direct contact with the destination. In this simulation I used 4-copy-4-hop for the MCP scheme which has been shown by experience (see Section 2.6) to be the best naive scheme in this kind of conference scenario in terms of delivery and cost.



(a) Delivery



(b) Cost

Figure 4.4: Comparison of delivery ratio and cost of different strategies

In Figure 4.4(a) we can see that, as expected, LABEL has a delivery ratio between MCP and

[6]Because of hardware problems, three out of eighty did not yield any data.

WAIT, and the trend is for it to approach closer to the performance of MCP, as the allowed time TTL of the messages increases.[7] In terms of cost, in Figure 4.4(b) we can see that MCP costs much more than LABEL, especially when the time TTL is increased up to 1 day, where MCP has less than a 10% improvement over LABEL, but it has around 6 times higher cost. Of course, WAIT has the lowest cost: since it is during a conference, it does not need to wait too long to meet the destination directly, hence the delivery ratio is not too low. Figure 4.5 shows the number of other nodes met directly by each node during the experimental period. It shows that almost every node has the chance to meet most of the other nodes during that period. However, if we look at the half-life delivery value, we can see that the half-life delivery is 3 hours for MCP, 9 hours for LABEL and around 1 day for WAIT. In other words, if you can tolerate a 1 day delay, you could use the WAIT strategy, otherwise LABEL would perform the best in terms of delivery ratio, delay and cost.

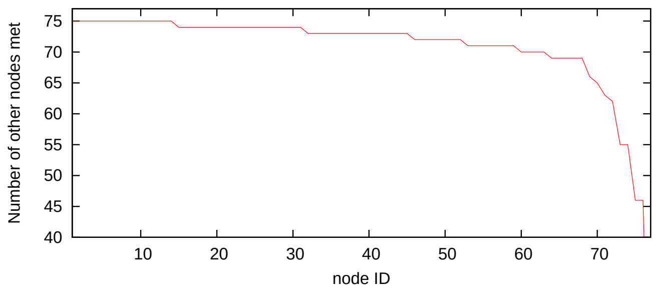


Figure 4.5: The number of other nodes met directly by each node during the experimental period.

The randomly generated groups scenario CONTROL is around 5% worse in terms of average delivery than LABEL in many cases, but it has a wider confidence interval than LABEL. (I also plotted the maximum and minimum bounds. And I find out that the minimum bounds of CONTROL are usually around 10% lower than the average value of the LABEL strategy.) This means that a badly generated combination of groups would affect the delivery quite significantly. Considering that 34 out of the 78 nodes have a community relationship, it is not difficult for the random groups generated to consist of members which belong to real affiliation groups. Furthermore, considering that during a conference, people from different research groups often mix together, this kind of performance is reasonable.

Figure 4.6 shows further the improvement of LABEL's performance compared to MCP in terms of delivery against cost. Each point represents a different time TTL value, and we can see that as we vary this, the delivery almost varies linearly with the cost. The anomaly after the cost is equal to 22 for the MCP case, where the slope slightly decreases is due to the fact that the system is going into saturation, and further increases in cost bring slower increases in delivery.

[7]Readers can observe that around 50% of the messages can be delivered within 3 hours using MCP. Although I have shown in Chapter 2 that for power-law coefficient, α, small than 1, expected delay is a day or even longer for all the datasets, if a system does not require expected delay guarantee for all packets (like this example), PSN is still possible.

Clearly LABEL has a much steeper slope than MCP: this means that this strategy is much more cost effective.

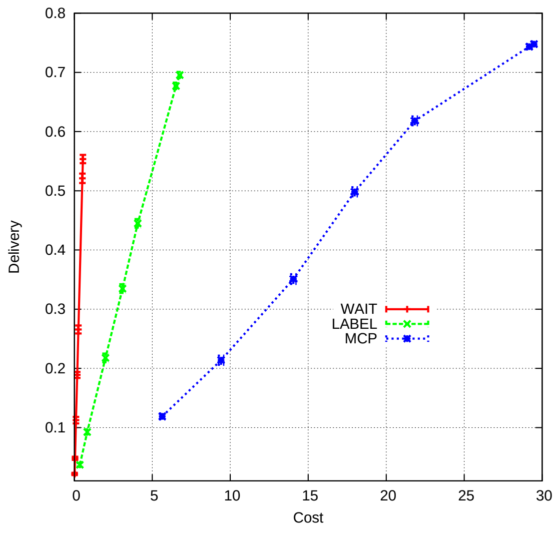


Figure 4.6: The Delivery-Cost Graph for MCP, LABEL and CONTROL strategies.

In Figure 4.7, I also look at the hop distribution, which is the distribution of the number of hops required for all the delivered messages. In this case, I set the time TTL to 3 hours as I have done in Section 2.6. The x-axis shows the number of hops and the y-axis shows the probability for a message to be delivered with at least that number of hops. Here I show the maximum and minimum bounds as well. For this value of time TTL, an maximum of 50% of the total messages created can be delivered (the y-value at 1-hop). For MCP, half of the messages delivered traversed 4 hops (the y-value at 4-hop minus the y-value at 5-hop), because MCP sends out messages on a blind first-come-first-send approach. But instead, for LABEL, the delivery ratio is almost the same for 1-hop and 4-hop, and slightly bigger in the 3-hop case. The direct contact case (1 hop) only helps to deliver less than 10% of the messages; much of the delivery relies on intermediate social relays.

In order to ascertain whether the friendship group concept is helpful for message delivery, I ran another series of simulations. In these, members of different friendship groups are allowed to act as relays for each other. I assigned the two groups from Paris to be friendship-groups of each other, and they help to relay messages. The result I expected to see is that the use of friendship groups can help to improve the delivery ratio, without too much increase in the delivery cost. A controlled experiment is also done, by using a random group chosen as a friendship group, rather than one with a known affiliation.

We can see from Figure 4.8(a) and Figure 4.8(b) that the friendship group did indeed help to improve delivery, with only slightly increased cost, Also, as expected, the randomly generated friendship group just increased the cost, without any improvement in delivery. It is difficult to study group and friendship group behavior in a conference, since the people are often mixing promiscuously: that is one of the purposes of a conference; but we can still make some

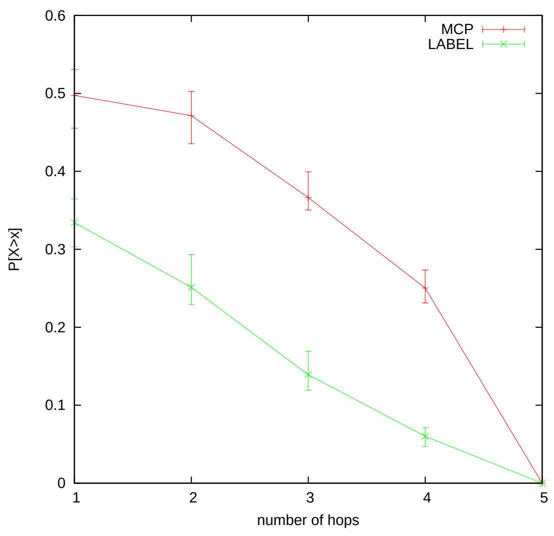
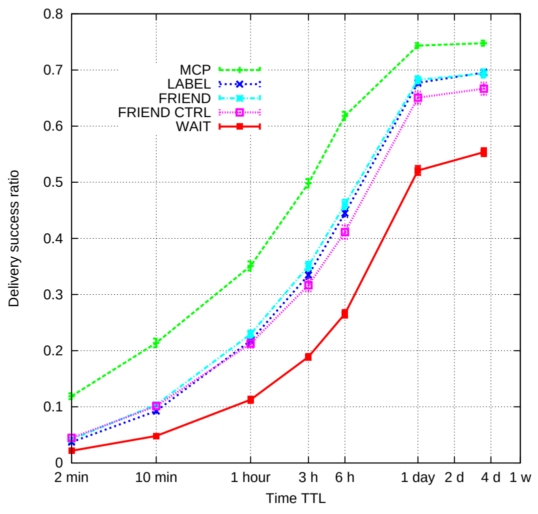
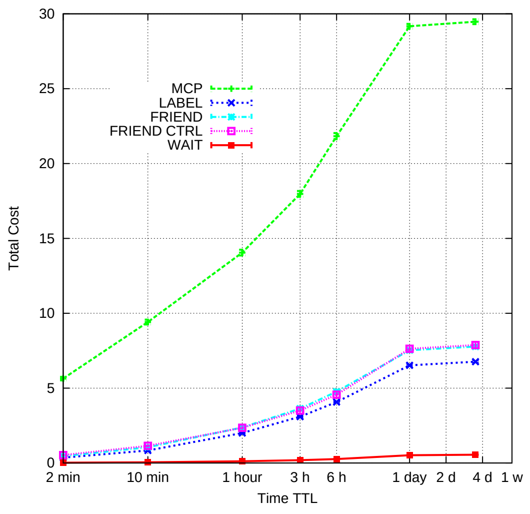


Figure 4.7: The hops distribution for all the deliveries in MCP and LABEL.



(a) Delivery



(b) Cost

Figure 4.8: Comparison of delivery ratio and cost on different strategies, with friendship groups.

observations about the correlation we see. I further believe that the techniques and metrics I have developed here can be used for research on friendship groups with more easily specified boundaries.

4.7 Modeling as Temporal Graphs

As *a priori* measurement work reported in Chapter 2 and Section 4.4, inter-contact time of humans follows a power-law distribution with a power-law coefficient smaller than 1. In this section, I introduce how PSNs can be modeled as temporal graphs with power-law distributed inter-contact time and also with nodes forming a community structure, as described in Chapter 3. There is still no empirical proof about the topology of a PSN, but usually communication networks are modeled as Erdős-Rényi random graphs [Bol01] (ER model) or scale-free graphs [CJW06]. Here I aim to model the underlying topology of PSNs by both of these two kinds of graphs. This section is not about forwarding, but an idea by combining community detection in the previous chapter and inter-contact time analysis in this chapter.

The modeling of PSNs here is divided into three phases, 1) the topological attachment phase, 2) the community detection phase, and 3) the temporal attachment phase. In the topological attachment phase, the topology of the graph is created by either the ER model or the Barabasi-Albert model [AB02] with the clustering size controlled by a linking probability. In the community detection phase, the graph created in phase one is split into communities. (some communities may only contain a single node.) In the third phase, inter-contact time distributions with different power-law coefficients are assigned to node-pairs, higher for within-community and lower for between communities.

4.7.1 Topological Attachment Phase

On Erdős-Rényi graphs, a network is characterised by two parameters: the size N and the link probability *p*. The mean degree is *z* = p(*N*-1). The percolation transition takes place at $p = p_c \equiv 1/N$, where p is the probability that two vertices are connected and *N* is the total number of vertices in the graph. The appearance of a giant component, which is also referred to as the percolating component, results in a dramatic change in the overall topological features of the graph.

In scale-free networks, some nodes act as ”highly connected hubs” (high degree), although most nodes are of low degree. Scale-free networks' structure and dynamics are independent of the system's size N, the number of nodes the system has. Their most distinguishing characteristic is that their degree distribution follows a power-law relationship, where the coefficient γ may vary approximately from 2 to 3 for most real networks. To model scale-free networks, I use the Barabasi-Albert model. The algorithm of the Barabasi-Albert model is the following:

(1) *Growth*: Starting with a small number (m_0) of nodes, at every time step, a new node is added with *m*($\ll m_0$) edges that link the new node to *m* different nodes already present in the system.

(2) *Preferential attachment*: When choosing the nodes to which the new node connects, the probability Π that a new node will be connected to node *i* depends on the degree k_i of node *i* is assumed, such that

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}, \tag{4.1}$$

After t time steps this procedure results in a network with $N = t + m_0$ nodes and *mt* edges.

The first step of building up the temporal graph is to build up the topological attachment. We can control the cluster sizes by varying the linking probability *p*. When $p < p_c$, the cluster size decays exponentially for large s:

$$P_p(|C| = s) \sim e^{-\alpha(p)s} : s \to \infty, \tag{4.2}$$

where $\alpha(p) \to \infty$ as $p \to 0$ and $\alpha(p_c)$=0. When $p > p_c$, the cluster size $P_p(|C| = s)$ follows a stretch exponential, $e^{-\beta(p)s^{(d-1)/d}}$, where d is the dimension of the lattice, but this dependence also vanishes as $d \to \infty$. By using these percolation thresholds, we can have a good reference to control the cluster sizes of the graphs we created.

4.7.2 Community Detection Phase

The second phase is to run the community detection algorithm to split the graphs from the previous stages into communities. I have tested the algorithm on 220 random graphs of 1000 vertices. The percolation threshold of a random graph of 1000 vertices is $1/1000 = 0.001$. I use 11 different linking probabilities, ranging from 0.0004 to 0.003 so we see the change before and after the percolation threshold, and for each linking probability I create 20 topologies. I measured the number of communities (num), maximum community size (S_{max}), average community size without counting the size-2 community (S_{wo2}), and average community size including the size-2 community (S_{wi2})for each graph; Figure 4.9(a) and 4.9(b) summarise all the results.

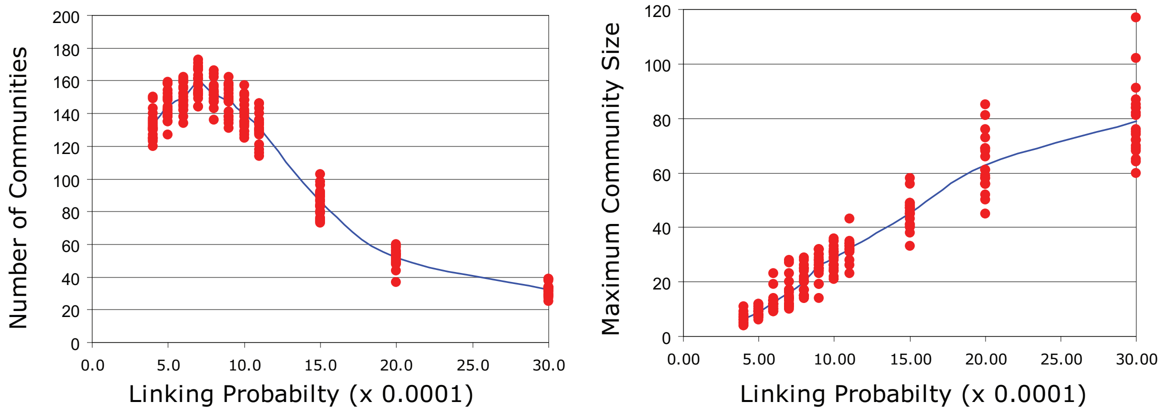


Figure 4.9: num(left) and S_{max}(right) against P_{link}

We can see that S_{max} increases almost linearly with the linking probability (P_{link}). This tells us that in order to increase the maximum community size of the population, we can just increase the linking probability. The number of communities, num, first increases with increasing P_{link} until a maximum and then decreases with P_{link}. When P_{link} is very low (e.g. 0.0004), the network created is very sparse and more nodes are not connected to any other nodes, and when P_{link} increases, the network becomes more and more connected and hence more clusters form. When P_{link} increases to a certain value, more big clusters form and hence fewer communities.

The average community size increases slowly and linearly until 0.001 and increases more rapidly after this threshold. This observation is similar for both S_{wo2} and S_{wi2}. And we can

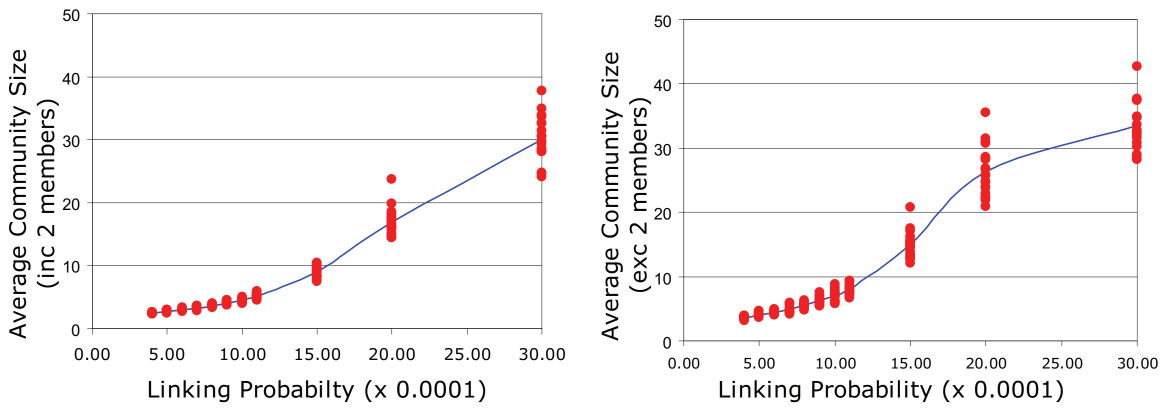


Figure 4.10: S_{wi2}(left) and S_{wo2}(right) against P_{link}

also see that S_{wo2} is quite different from S_{wi2} until $P_{link} = 0.002$. This indicates that quite a significant amount of communities are size-2 communities. These graphs give us some guidelines on how to use the linking probability to estimate the average community size, the number of communities and also maximum community size of the graphs to be created, and also give us some information about the working performance of this community detection algorithm. Because this is not a main focus of the dissertation, I will leave the community detection of random graphs with higher linking probability and scale-free graphs as future work. After this phase, we can move to create the temporal attachment for the node pairs.

4.7.3 Temporal Attachment Phase

As measured in [CHC+06] and [HCS+05], the inter-contact time of a node pair follows a power-law distribution. Here I will show empirically that the power-law coefficients are quite different for node pairs within communities and between communities. Figure 4.11 shows the inter-contact time distribution extracted from a typical node with nodes within the same community and also nodes in another community in the *Cambridge* dataset. The two graphs contain the same number of node pairs. We can see that for the within-community case, the inter-contact time between 800 seconds and 10000 seconds can be approximated by a power-law distribution with power-law coefficient varying between α_{low} and α_{up}. We can also see from the right figure that the number of contacts significantly decreases between communities. This results in many discrete steps. We also observed fewer lines in the between-community figure even though they contain the same number of node pairs. This is because some nodes have no contacts with that node. Similar observations for within-community and between-community inter-contact distributions are also found for other node pairs in the same dataset and also other datasets. This gives us hints on how to model the temporal attachment of the nodes.

After the above community detection phase, the random or scale-free graphs generated are split into communities. Then I assign a series of inter-contact times for each node pair. For a within community pair, the power-law coefficient of its inter-contact time distribution, α, is picked uniformly between a predefined upper bound α_{up} and α_{low}, that is $\alpha_{low} \leq \alpha \leq \alpha_{up}$. And for node pairs between communities and also for nodes not belonging to communities,

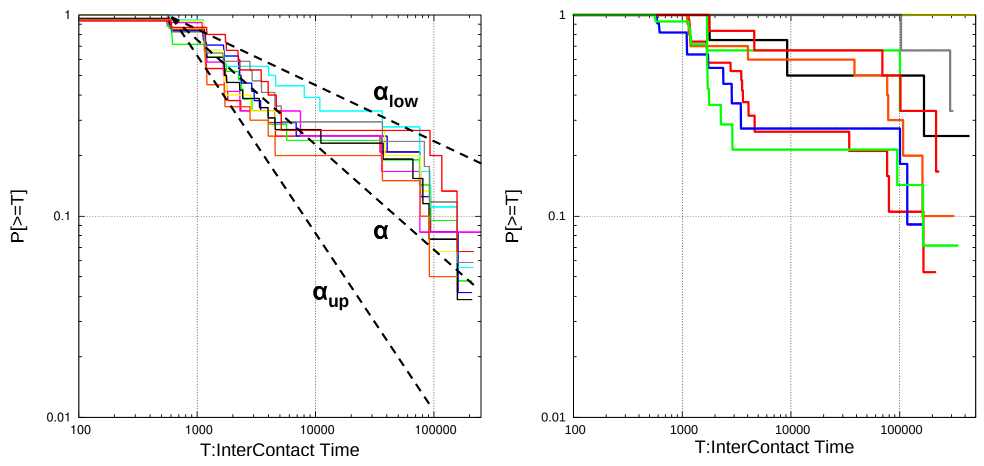


Figure 4.11: Inter-contact distribution, within group (left) and between group (right),*Cambridge*

their power-law coefficients of inter-contact time distribution are picked uniformly between 0 and $(\alpha_{up} + \alpha_{low})/2$. Here all the node pairs I mentioned above are all connected nodes on the random graphs created in the topological attachment phase, so they should have at least one contact. After this phase, the temporal graphs are ready to use.

The limitation of my model is that it is contact-based and does not incorporate spatial information, so it cannot be used to test spatial-based algorithms i.e. landmark routing. But it did incorporate the two main features of human interaction measured in the literature, structurally local cohesive and temporal preferential attachment, and will be a useful step for better mobility modeling. Instead of random graphs and power-law graphs, I want to use real online social network topologies at the topological attachment phase in the near future, which may reflect better real human community structures [MMG+07]. Although this section is not on the main track of forwarding algorithm design, it provides early ideas on modeling and generating human mobility traces, which can be used to test forwarding algorithms. Hence it is related and I include it here for the readers' interest.

4.8 Conclusions

The addition of even a small amount of state information, i.e. in my experiments an affiliation label used to choose nodes preferentially, is shown to bring significant improvement in forwarding performance over oblivious or naive forwarding algorithms in PSN. This is the first empirical result of evaluating groups with *a priori* information. One experimental dataset may not be able to draw us a very general conclusion, but this is all I have with *a priori* labeling. One way to further confirm this result is to evaluate it on more experimental datasets with the communities detected by the algorithms in Chapter 3.

Another problem with the simple LABEL is that the source needs to wait until the community members of the destination or the destination are at one-hop distance from itself to start the forwarding. This may be not acceptable in many situations, since it may happen that the sender will never be in reach of a member of the community of the message recipient. We need to consider more efficient ways of moving the messages away from the source so we can combine several ways together to improve the overall performance. In the next chapter, I look into these two problems.

Chapter 5

Social Based Forwarding in Small World DTNs

In this paper I seek to understand human social context at multiple levels of detail and use it in the design of forwarding algorithms for Pocket Switched Networks (PSNs). From human mobility traces taken from the real world, I discover the heterogeneity of human interaction, including communities and hubs. Society naturally divides into communities, and individuals have varying popularity. I propose a social based forwarding algorithm, BUBBLE, which is shown empirically to improve forwarding efficiency significantly. This is a follow-up chapter based on the community detection methodologies in Chapter 3 and the LABEL strategy in Chapter 4.

5.1 Introduction

The *first* generation of human network models were probably the Erdős-Rényi random graphs [Bol01]. More recently, heterogeneity has been introduced into models through the use of power-law and small-world graphs, especially in analysis of the AS-level of the Internet [CJMW05] [CJW06]. This is the *second* generation of modeling. It is well known that some nodes may be more highly connected to each other than to the rest of the network. The set of such nodes are usually called clusters, communities, cohesive groups or modules. Many different approaches to community detection in complex networks have been proposed such as K-CLIQUE [PDFV05], betweenness [NG04], modularity [New06] and more recently methods based on information theory [RB06b] and statistical mechanics [RB06a]. Other methods can be found in survey papers [New04b] [DDDGA05]. Community detection can help us understand the local structure in mobility traces, and therefore help us design good strategies for information dissemination. It may be that communities detected from mobility data do not actually match well to real social communities, but still help with improved forwarding. [1]

[1] I have found out that actually they match quite well in Chapter 3.

The first goal of this chapter is to move to a third generation of human mobility models, understanding heterogeneity at multiple levels of detail.

Our previous work [CHC$^+$06] (Chapter 2) established inter-contact intervals, and contact durations for a wide range of typical human mobility patterns and for a variety of today's radio devices. Critically, it was shown that stateless forwarding schemes would not provide a bounded expected mean delivery latency across such systems. On the other hand, flooding packets has a very high cost, not just in link-utilisation, but for other resources such as node storage and battery life, which are likely to be highly valued by users.

The second goal of this chapter is to devise efficient forwarding algorithms for PSNs which take advantage of both **a priori** *and learned knowledge of the structure of human mobility, to provide improved performance trade-of between delivery probability, latency and cost.*

In this chapter, I focus on two specific aspects of society: community and centrality. *Community* is an important attribute of PSNs. Cooperation binds, but also divides human society into communities. Human society is structured. Within a community, some people are more popular, and interact with more people than others (i.e.have high *centrality*); we call them hubs. Popularity ranking is one aspect of the population. As I have introduced the correlated interaction in the last chapter, an organism of a given type is more likely to interact with another organism of the same type than with a randomly chosen member of the population [Oka05]. This correlated interaction concept also applies to human, so we can exploit this kind of community information to select forwarding paths. To date, though, there have been few results to support this conjecture that I am aware of, except my very preliminary analysis on the use of as users' affiliation [HC07]. Betweenness centrality measures the number of times a node falls on the shortest path between two other nodes [Fre77]. This concept is also valid in a temporal network. In a PSN, it can represent the importance of a node for relaying traffic for others in the system. Hence, I will look at whether hierarchical search works with this centrality metric, and how to acquire the metric in a practical, decentralised way.

I evaluate the impact of community and centrality on forwarding and propose a hybrid algorithm, BUBBLE, that uses both. I demonstrate a significant improvement in forwarding efficiency. I focus on empirical analysis; I do not consider abstracting a mathematical model in this work, but evaluate the forwarding schemes directly on the mobility traces.

There are five specific contributions in this chapter that progress towards my two top-level goals. First, I explore human heterogeneity in the datasets (Section 5.2). Second, I show empirically that identifying nodes according to their centrality or ranking can improve delivery cost-effectiveness over a greedy forwarding approach (Section 5.4). Third, I demonstrate the limitations of the LABEL algorithm (introduced in Chapter 4) of solely using community information (Section 5.5). Fourth, I combine community and centrality together, making use of both local and global structures. This reduces the dead-end effect caused by global ranking, by forming a hybrid forwarding strategy, BUBBLE, which improves over the delivery performance of naive multiple-copy-multiple-hop (MCP) flooding schemes and PROPHET [LDS04], but with much lower cost (Section 5.6). Finally, I use average unit-time degree to approximate centrality, and show that this achieves nearly the same performance as greedy ranking (Section 5.7).

5.2 On Human Heterogeneity

The human heterogenous structures I want to explore are community (heterogeneity in cohesiveness) and hubs (heterogeneity in centrality). Cohesiveness indicates the local clustering and centrality identifies the importance of the role of a node in the network.

5.2.1 Experimental Datasets

I use experimental datasets gathered by my for a period of 2 years referred to as *Hong Kong*, *Cambridge*[2], *Infocom05*, *Infocom06*, and one other dataset from the MIT Reality Mining Project [EP06], referred to as *Reality*. Previously the characteristics of these datasets such as inter-contact and contact distribution, have been explored in several studies [CHC$^+$06] [HCS$^+$05] [LLS$^+$06] and also Chapter 2 of this thesis, to which I refer the reader for further background information.

5.2.2 Heterogeneity in Cohesiveness

As I studied in Chapter 3, the participants among all these experiments form local clusters or community structures. This community structure is an important characteristic to be considered in human mobility modeling, forwarding algorithms, and applications to design for PSNs. In Chapter 4, I have done an early study on using affiliation information to improve forwarding efficiency in the conference scenario. In this chapter, I will verify it in different environments and hence to draw more general conclusions.

There are two features about human communities to emphasise here. One is the overlapping characteristic and the other is the hierarchical characteristic. As I analysed the data using K-CLIQUE community detection, I found out that some of the communities overlap each other. One person may belong to multiple communities, and bridge data or epidemics from one community to another community. So it is important to consider and identify this kind of bridge node in algorithms. Also when we increased the threshold values for the detection criteria, we observed the shrinking in the community sizes and some big communities further split into smaller stand alone communities. This is the hierarchical nature of human communities. I will show how I can use these two features in my forwarding design in the later sections.

[2]Here, *Cambridge*, I refer to *Cambridge05* in Section 2.3.

5.2.3 Heterogeneity in Centrality

In many mobility models such as random way point, nodes are assumed, explicitly or implicitly, to have homogeneous speed distributions, importance and popularity. My intuition is that the last two assumptions, at least, are not true. People have different levels of popularity: salesmen and politicians meet customers frequently, whereas computer scientists may only meet a few of their colleagues once a year. Homogeneity might favour different forwarding strategies for PSNs. In contrast, I want to employ heterogeneous popularity to help design more efficient forwarding strategies: I prefer to choose popular hubs as relays rather than unpopular ones. To date I am not aware of any empirical evidence for using human popularity or node centrality for information dissemination in mobile networks.

A temporal network is a kind of weighted network. The centrality measure in traditional weighted networks may not work here since the edges are not necessarily concurrent (i.e. the network is dynamic and edges are temporal-dependent). Hence we need a different way to calculate the centrality of each node in the system. My approach is as follows:

1. Carry out a large number of emulations of unlimited flooding with different uniformly distributed traffic patterns created using the *HaggleSim* emulator.

2. Count the number of times a node acts as a relay for other nodes on all the shortest delay deliveries. Here the shortest delay delivery refers to the case when the same message is delivered to the destination through different paths, where I only count the delivery with the shortest delay.

I call this number calculated above the *betweenness centrality* of this node in this temporal graph[3]. Of course, it can be normalised to the highest value found. Here I use unlimited flooding since it can explore the largest range of delivery alternatives with the shortest delay. This definition captures the spirit of Freeman centrality [Fre77].

Initially, I only consider the homogeneous communication pattern, in the sense that every destination is equally likely, and I do not weight the traffic matrix by locality. I then calculate the global centrality value for the whole homogeneous system. Later, I will analyse the heterogeneous system (Section 5.6).

Figure 5.1 shows the number of times a node falling on the shortest paths between all other node pairs. We can simply treat this as the centrality of a node in the system. I observed a very wide heterogeneity in each experiment. This clearly shows that there is a small number of nodes which have extremely high relaying ability , and a large number of nodes have moderate or low centrality values, across all experiments. One interesting point from the HK data is that the node

[3] I have calculated the weighted node centrality for each node, but found out that the weighted centrality is not well correlated to the centrality on the temporal graph. Nodes having high static weighted centrality may have low temporal centrality.

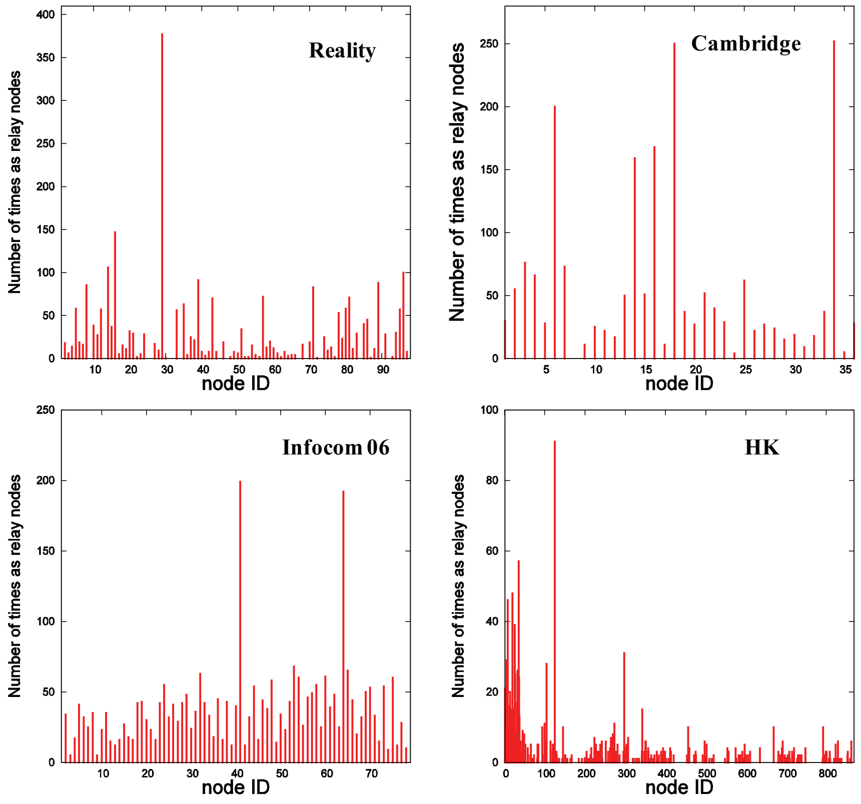


Figure 5.1: Number of times a node as relays for others on four datasets.

showing highest delivery power in the figure is actually an external node. This node could be some popular hub for the whole city, i.e.postman or a newspaper man in a popular underground station, which relayed a certain amount of cross city traffic. The 30th, 70th percentiles and the means of normalised individual node centrality are shown in Table 5.1 and the distributions are show in Figure 5.2.

Experimental dataset	30th percentile	Mean	70th percentile
Cambridge	0.052	0.220	0.194
Reality	0.005	0.070	0.050
Infocom06	0.121	0.188	0.221
Hong Kong	0.000	0.017	0.000

Table 5.1: Statistics about normalised node centrality in 4 experiments

5.3 Interaction and Forwarding

In the first half of this chapter and in Chapter 3, I have shown the existence of heterogeneity at the level of individuals and groups, in all the mobility traces. This motivates us to consider a new heterogeneous model of human interaction and mobility.

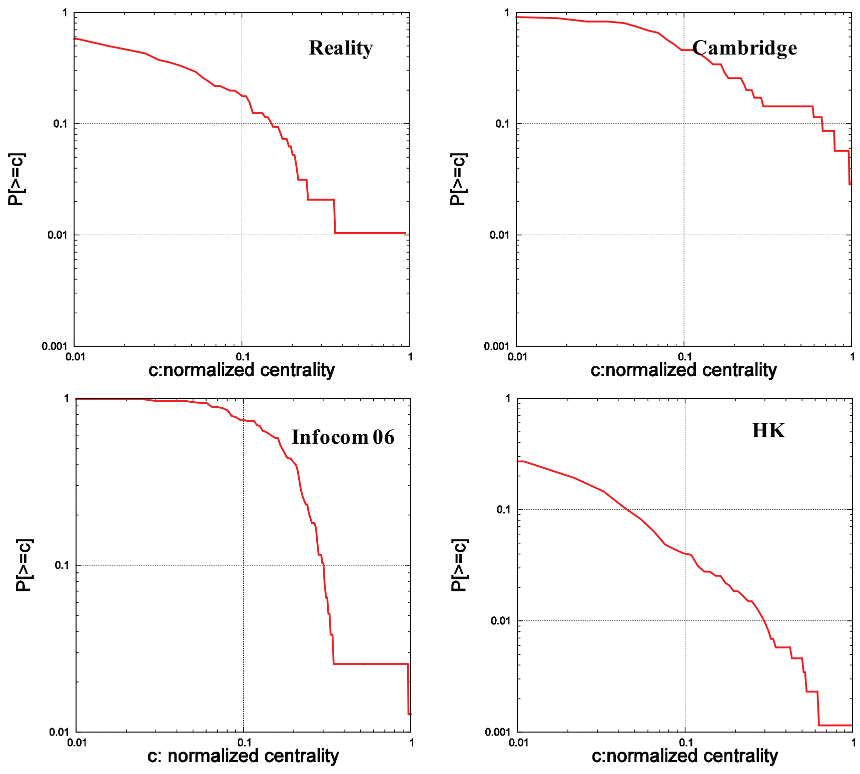


Figure 5.2: Distribution of normalised node centrality on four datasets.

Categories of human contact patterns Human relationships can be modeled by using correlation of contact duration and number of contacts. I defined four types of human relationship based on the correlation of contact duration and number of contacts (Section 3.3).

Cliques and Community I explored the community structures inside different social environments, and found these community structures match quite well with the real underlying social structures (Section 3.4 and 3.5).

Popularity Ranking We shall see that popular hubs are as useful in the PSN context as they are in the wireline Internet and in the Web.

I also provide the details of the statistics of interactions for the experiments (see Chapter 3) so that they can be used by other researchers in future modeling, or to bootstrap larger experiments of composites of these.

From Section 5.4 to Section 5.7, I look at how can we use this information to make smart forwarding decisions. The following three pre-existing schemes provide lower and upper bounds in terms of cost and delivery success. All of these schemes are inefficient because they assume a homogeneous environment. If the environment is homogeneous then every node is *statistically equivalent* (i.e. every node has the same likelihood of delivering the messages to the destination). As I showed in the first half of this chapter, the environments and nodes are diverse, and hence all these naive schemes are doomed to have poor performance. We need to design algorithms which make use of this rich heterogeneity.

- **WAIT** Hold onto a message until the sender encounters the recipient directly, which represents the lower bound for delivery and cost.
- **FLOOD** Messages are flooded throughout the entire system, which represents the upper bound for delivery and cost.
- **MCP** Multiple-Copy-Multiple-Hop. Multiple Copies are sent subject to a time-to-live hop count limit on the propagation of messages. By exhaustive emulations, the 4-copy-4-hop MCP scheme is found to be the most cost-effective scheme in terms of delivery ratio and cost for all naive schemes among all the datasets except the *HongKong* data (see also Section 2.6). Hence for fair comparison, I would like to evaluate my algorithms against the 4-copy-4-hop MCP scheme in most of the cases.

The Mobile network has a dual nature: it is both a physical network and at the same time it is also a social network. A node in the network is a mobile device, and also associated with a mobile human.

Figure 5.3 shows the design space for the forwarding algorithms in this chapter. The vertical axis represents the explicit social structure, that is facets of nodes that can specifically identified such as affiliation, organisation or other social context. This is the social or human dimension. The two horizontal axes represent the network structural plane, which can be inferred purely from observed contact patterns. The Structure-in-Cohesive Group axis indicates the use of localised cohesive structure, and the Structure-in-Degree axis indicates the use of hub structure. These are observable physical characteristics. In my design framework, is not necessary that physical dimensions are orthogonal to the social dimension, but since they are represent two different design parameters, I would like to separate them. The design philosophy here is to consider both the social and physical aspects of mobility.

I introduce four forwarding algorithms in this chapter, namely LABEL, RANK, DEGREE, and BUBBLE.

LABEL Explicit labels are used to identify forwarding nodes that belong to the same organisation. Optimisations are examined by comparing label of the potential relay nodes and the label of the destination node.This is in the human dimension, although an analogous version can be done by labeling a k-clique community in the physical domain.

RANK This is analogous to the degree of a node in a fixed network; I use a modified ranking scheme, namely the node centrality in a temporal network. A message is forwarded to nodes with higher centrality values than the current node. It is based on observations in the network plane, although it also reflects the hub popularity in the human dimension.

DEGREE A heuristic based on the observed average of the degree of a node over some longer interval. Either the last interval window (S-Window), or a long-term cumulative estimate, (C-Window) is used to provide a fully decentralised approximation for each node's centrality, and then that is used to select forwarding nodes.

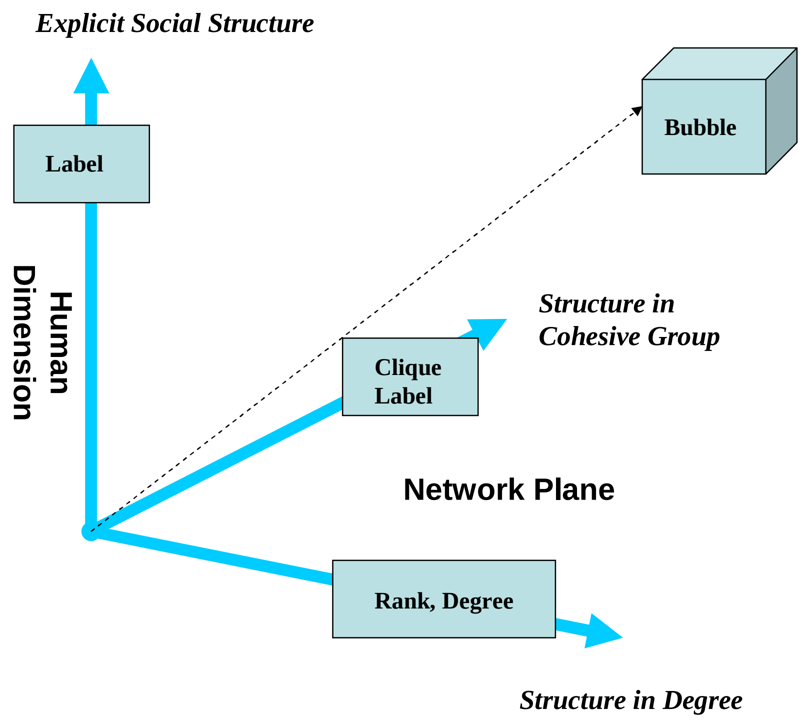


Figure 5.3: Design space for forwarding algorithms.

BUBBLE The BUBBLE family of protocols combines the observed hierarchy of centrality of nodes with explicit labels, to decide on the best forwarding nodes. BUBBLE is an example algorithm which uses information from both human aspects and also the physically observable aspects of mobility.

BUBBLE is a combination of LABEL and RANK. It uses RANK to spread out the messages and uses LABEL to identify the destination community. For this algorithm, I make two assumptions:

- Each node belongs to at least one community. Here I allow single node communities to exist.
- Each node has a global ranking (i.e. global centrality) in the whole system and also a local ranking within its community. It may belong to multiple communities and hence may have multiple local rankings.

In the following sections, I will show how can we make use of these different metrics to improve forwarding performance in a heterogeneous system and also when they will fail.

5.4 Greedy Ranking Algorithm

The third contribution of this chapter is to modify the greedy ranking search scheme over power-law networks to apply to temporal graphs, and evaluate the resulting algorithm.

5.4.1 The Power of Greedy Ranking

Here I use a similar greedy strategy to the one Adamic *et al.* introduced in [AHLP01]. A PSN is not a static network like the Internet: we do not know when a local maximum is reached since the next encounter is unexpected. We cannot employ precisely the same strategy as they propose. Here I assume each node knows only its own ranking[4] and the rankings of those it encounters, but does not know the ranking of other nodes it does not encounter, and does not know which node has the highest rank in the system. My strategy, which I call RANK, is very simple: we keep pushing traffic on all paths to nodes which have a higher ranking than the current node, until either the destination is reached, or the messages expire.

If a system is small enough, the global ranking of each node is actually the local ranking. If we consider only the Systems Research group (around 40 people), a subset of the Cambridge Computer Laboratory (235 people), this is the ranking of each node inside the group. If we consider the whole Computer Laboratory, we are considering a larger system of many groups, but they all use the same building. A homogeneous ranking can still work. But when we consider the whole city of Cambridge, a homogeneous ranking system would exclude many small scale structures. In this section, I show that in relatively small and homogeneous systems, a simple greedy ranking algorithm can achieve good performance.

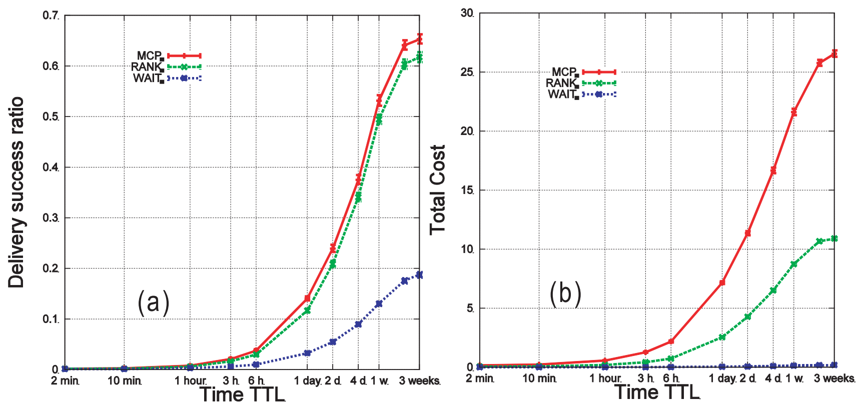


Figure 5.4: Comparison of delivery ratio (left) and cost (right) of MCP and RANK on 4 copies and 4 hops case (*Reality*).

Figure 5.4(a) shows that the simple greedy ranking performs almost as well as MCP for delivery.[5] Figure 5.4(b) also shows that the cost is only around 40% that of MCP, which represents a marked improvement.

Hierarchical organisation is a common feature of many complex systems. The defining feature of a hierarchical organisation is the existence of a hierarchical path connecting any two of its

[4]A node can know its own ranking from a central server or using distributed approximations, which I will show in Section 5.7.

[5]In general, most of the deliveries experience a large latency in the *Reality* dataset (e.g. take up to one week to deliver 50% of the created messages). The reason is that this dataset is quite sparse. Some participants may switch off their Bluetooth radios sometimes to save power of their phones.

nodes. Trusina *et al.* [TMMS04] address how to detect and measure the extent of the hierarchy manifested in the topology of a given complex network. They defined the hierarchical path based on node degrees. A path between two nodes in a network is called hierarchical if it consists of an *up path* where one is allowed to step from node i to node j only if their degrees k_i, k_j satisfy $k_i \leq k_j$, followed by a *down path* where only steps to nodes of lower or equal degree are allowed. Either the up or down path is allowed to have zero length. Because of the good results from the greedy ranking algorithm, I analysed the percentage of hierarchical paths inside all the shortest paths. Table 5.2 summarises the results.

Experimental dataset	% hierarchical paths
Cambridge	87.2 (-2.4,+4.3)
Reality	81.9 (-3.1,+3.3)
Infocom05	62.3 (-2.5,+2.5)
Infocom06	69.5 (-4.1,+2.4)
Hong Kong	33.5 (-4.0,+4.0)

Table 5.2: Hierarchical paths analysis of all shortest paths

The percentage of hierarchical paths is calculated as the number of hierarchical paths divided by the number of non-direct deliveries. We can see that for *Cambridge* data and *Reality*, the percentage of hierarchical paths is very high, so our strategy of pushing the messages up the ranking tree can find a lot of these paths, and the performance of the ranking strategy here is not much different from that of MCP. For *Infocom06* and *Infocom05*, the percentage of hierarchical paths is also high, hence the greedy RANK strategy can as well discover many of the shortest paths. However, for the *Hong Kong* experiment, the network is too sparse and a lot of shortest paths are hidden. (This occurs because I could not know the devices detected by the external devices, and most of the resulting paths used for delivery are actually not the shortest) We can see that percentage of hierarchical paths controls the delivery success achieved by the greedy RANK algorithm. I conclude from this that a high percentage of the shortest paths are actually hierarchical paths.

5.4.2 When the Greedy Ranking Fails

RANK appears to work in small and homogeneous systems, but when we look at a more diversified system, for example the *Hong Kong* dataset, it may work differently. In the *Hong Kong* experiment, the 37 participants are intentionally selected without any social correlation. They live and work throughout the whole city. Relying on direct contact, less than 4% of the messages can be delivered.[6] Unlike what I did to all the previous datasets, here all the external

[6]The confidence interval in this case is similar to the *Reality* dataset. I show the maximum and minimum instead to demonstrate the upper and the lower bounds for this dataset.

Bluetooth devices detected need to be used for constructing the paths. But, because we do not know the devices detected by all these external devices, a lot of potential paths were not found.

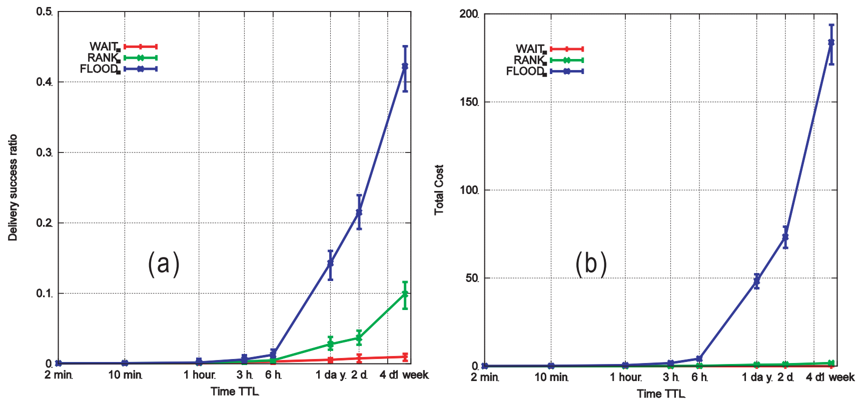


Figure 5.5: Comparison of delivery ratio and cost of MCP and Greedy RANK on no constraints case (HK)

Figure 5.5 shows the delivery ratio and delivery cost using flooding, and using uncontrolled greedy ranking (i.e. not controlled by number of hops and copies). We can see that using flooding, we can deliver more than 40% of the total traffic across the whole city by using only the 37 iMotes and the external devices detected by these iMotes. Without knowing the devices detected by the external devices, that will be a huge number of paths out of these 869 devices. However the cost is also very high: to deliver one message, we need to make around 180 copies. But in this case, greedy ranking can only deliver 10% of the messages, although the cost is much lower as well. In terms of delivery and cost, greedy ranking is still more cost-effective than flooding, but clearly the delivery success rate is still too low. One explanation for this low performance is that since the participants have no social correlation, and belong to different social communities, high global ranking of a node may not represent a good choice of relay for some local communities. Messages keep being pushed up to some globally higher ranking nodes, and getting stuck at some maxima, rather than then trickling down to some local community. Figure 5.6(a) shows that the maximum number of hops for greedy Rank is 4 hops and after that the messages get stuck. Figure 5.6(b) shows the rank distribution of the sources, destinations and dead-ends of all the undelivered messages, indicating that message delivery has typically failed at highly-ranked nodes. This supports my hypothesis concerning the dilemma of the messages getting stuck at maxima.

5.5 Direct Labeling Strategy

In the LABEL strategy [HC07] in Chapter 4, each node is assumed to have a label that tells others its affiliation, just like a name badge in a conference. The direct LABEL strategy refers to the exclusively using of labels to forward messages to destinations: next-hop nodes are selected

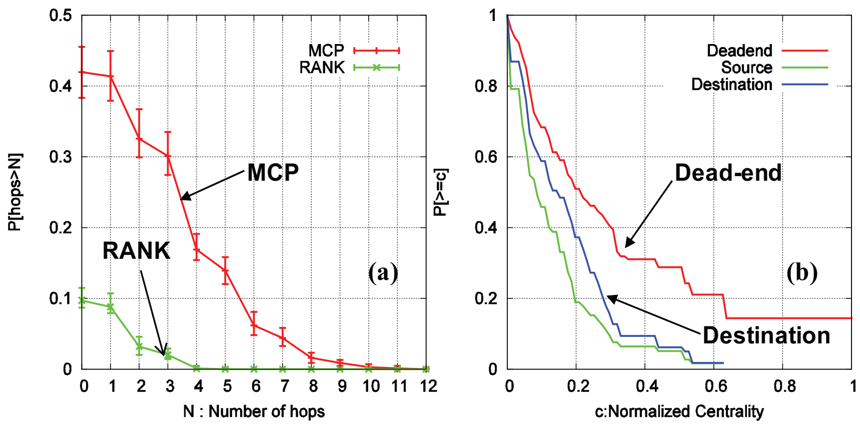


Figure 5.6: The hop distribution of the delivered (left) and the rank distribution of undelivered (right) on HK data.

if they belong to the same group (same label) as the destination. It was demonstrated that LABEL significantly improves forwarding efficiency over "oblivious" forwarding using *Infocom06* dataset. This is a beginning of social based forwarding in PSN, but lack of mechanisms to move messages away from the source when the destinations are socially far away (such as *Reality*). My third contribution is to demonstrate the limitations of LABEL strategy and move to a new forwarding algorithm using both community and centrality information.

5.5.1 The Problem with Direct Labeling

As I mentioned before, for all the datasets we have, only *Infocom06* dataset has *a priori* affiliation labels. But fortunately, a human community also represents one type of long term, stable relationship. An outside observer of human society would not know at first to which group each person belongs. As time goes by, we gain higher confidence concerning who usually socialises with whom. In this part of the analysis, I use the communities detected from the nine month *Reality* traces. Nine months is a long enough period for us to have high confidence to believe that the communities extracted from the dataset truly reflect the social communities existing between the participants. I evaluate the LABEL strategy on this dataset.

We can see from Figure 5.7 that LABEL only achieves around 55% of the delivery ratio of the MCP strategy and only 45% of the flooding delivery although the cost is also much lower. However, it is not an ideal scenario for LABEL. In this environment, people do not mix as well as in a conference. A person in one group may not meet members in another group so often, so waiting until the members of the other group appear to do the transmission is not effective here.

Figure 5.8 shows the correlation of the nth-hop relay nodes to the source and destination groups for the messages on all the shortest paths, that is the percentage of the nth-hop relay nodes that are still in the same group as the source or already in the same group as the destination. We can see that more than 50% of the nodes on the first hops (from the S-Group plot) are still in the same group as the source group of the message and only around 5% of the first hop nodes (from

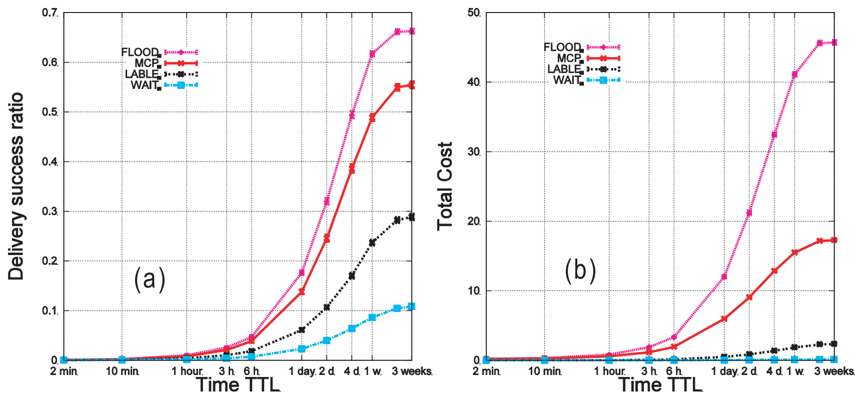


Figure 5.7: Comparison of delivery ratio (left) and cost (right) of MCP and LABEL on 4 copies and 4 hops case (*Reality*).

the D-Group plot) are in the same group as the destination. This explains why direct labeling is not effective, since it is far from discovering the shortest path. We can also see that on going to the 2nd hop, S-Group correlation drops to slightly less than 30%, and when going to 4th-hops, almost all (90%) messages have escaped from this source group. To calculate the percentage for each hop I divide the number of messages which belong to that group (S-Group or D-Group) by the total number of messages destined beyond nodes at that particular hop, but not the total messages created. In the 4-hop case, there are perhaps only 100 messages to forward further, and only 10 out of these 100 relay nodes belong to the source group. This explains why LABEL is not effective, since it is far from discovering the shortest path. In the next section, I will talk about how to use centrality to improve the delivery ratio of LABEL.

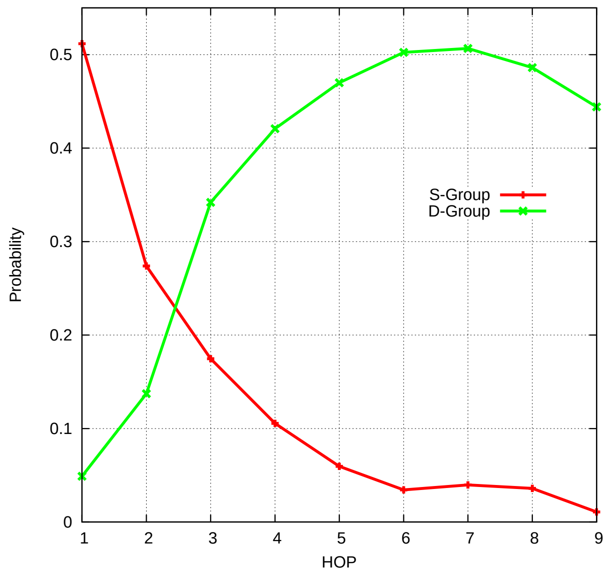


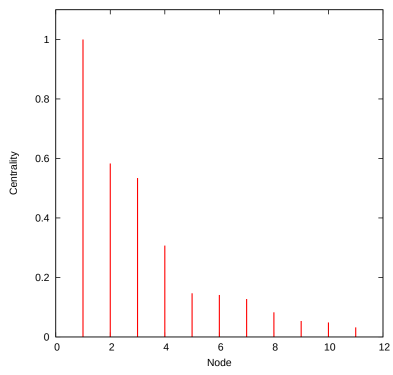
Figure 5.8: Correlation of nth-hop nodes with the source group and destination group (*Reality*).

5.6 Centrality Meets Community

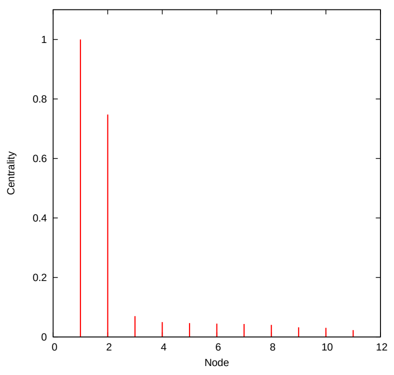
The fourth contribution in this chapter is to combine the knowledge of both the centrality of nodes and the community structure, to achieve further performance improvements in forwarding. I show that this avoids the occurrence of the dead-ends encountered with pure global ranking schemes. I call the protocols here BUBBLE, to capture our intuition about the social structure. Messages bubble up and down the social hierarchy, based on the observed community structure and node centrality, together with explicit label data. Bubbles represent a hybrid of social and physically observable heterogeneity of mobility over time and over community.

5.6.1 Two-community Case

In order to make the study more systematic, I start with the two-community case. I use the *Cambridge* dataset for this study. By experimental design, and confirmed using my community detection algorithm, we can clearly divide the *Cambridge* data into two communities: the undergraduate year-one and year-two group. In order to make the experiment more fair, I limit myself to just the two 10-clique groups found with a number-of-contact threshold of 29; that is where each node at least meet another 9 nodes frequently. Some students may skip lectures and cause variations in the results, so this limitation makes my analysis yet more plausible.



(a) Group A

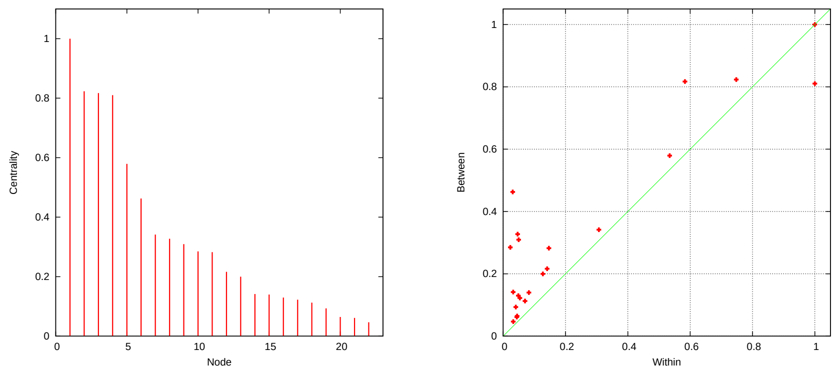


(b) Group B

Figure 5.9: Node centrality in 2 groups in *Cambridge* data

First I look at the simplest case, for the centrality of nodes within each group. In this case, the traffic is created only for members within the same community and only members in the same community are chosen as relays for messages. We can clearly see from Figures 5.9(a) and 5.9(b) that inside a community, the centrality of each node is different. In Group B, there are two nodes which are very popular, and have relayed most of the traffic. All the other nodes have low centrality value. Forwarding messages to the popular nodes would make delivery more cost effective for messages within the same community.

Then I consider traffic which is created within each group and only destined for members in another group. To eliminate other outside factors, I use only members from these two groups as relays. Figure 5.10(a) shows the individual node centrality when traffic is created from one group to another. Figure 5.10(b) shows the correlation of node centrality within an individual group and inter-group centrality. We can see that points lie more or less around the diagonal line. This means that the inter- and intra- group centralities are quite well correlated. Active nodes in a group are also active nodes for inter-group communication. There are some points on the left hand side of the graph which have low intra-group centrality but moderate inter-group centrality. These are nodes which move across groups. They are not important for intra-group communication but can perform certainly well when we need to move traffic from one group to another.



(a) Centrality (b) Correlation

Figure 5.10: Inter-group centrality and correlation between intra- and inter-group centrality (*Cambridge*)

I can show now why homogeneous global ranking in Section 5.4 does not work perfectly. Figure 5.11 shows the correlation of the local centrality of Group A and the global centrality of the whole population. We can see that quite a number of nodes from Group A lie along the diagonal line. In this case the global ranking can help to push the traffic toward Group A. However the problem is that some nodes which have very high global rankings are actually not members of Group A, for example node D. Just as in real society, a politician could be very popular in the city of Cambridge, but not a member of the Computer Laboratory, so may not be a very good relay to deliver message to the member in the Computer Laboratory. Now we assume there is a message at node A to deliver to another member of Group A. According to global ranking, we would tend to push the traffic toward B, C, D, and E in the graph. If we pushed the traffic to node C, it would be fine, and to node B it would be perfect. But if it push the traffic to node D and E, the traffic could get stuck there and not be routed back to Group A. If it reaches node B, that is the best relay for traffic within the group, but node D has a higher global ranking than B, and would tend to forward the traffic to node D, where it would probably get stuck again. Here I propose the BUBBLE algorithm to avoid these dead-ends.

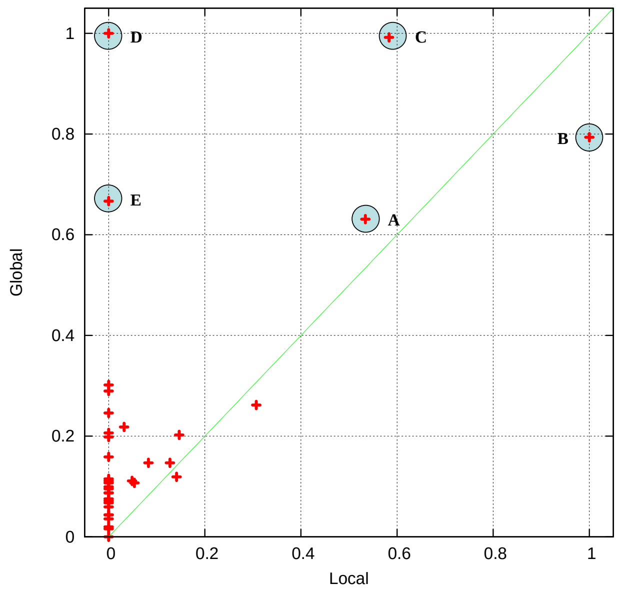


Figure 5.11: Correlation of local centrality of group A and the global centrality (*Cambridge*).

Algorithm 2: BUBBLE RAP

```
begin
    foreach $EncounteredNode\_i$ do
        if (LabelOf($currentNode$) == LabelOf($destination$)) then
            if (LabelOf($EncounteredNode\_i$) == LabelOf($destination$))
                and
            (LocalRankOf($EncounteredNode\_i$) > LocalRankOf($currentNode$)) then
                $EncounteredNode\_i$.addMessageToBuffer($message$)
        else
            if (LabelOf($EncounteredNode\_i$) == LabelOf($destination$))
                or
            (GlobalRankOf($EncounteredNode\_i$) > GlobalRankOf($currentNode$)) then
                $EncounteredNode\_i$.addMessageToBuffer($message$)
end
```

Forwarding is carried out as follows. If a node has a message destined for another node, this node would first bubble this message up the hierarchical ranking tree using the global ranking until it reaches a node which has the same label (community) as the destination of this message. Then the local ranking system will be used instead of the global ranking and continue to bubble up the message through the local ranking tree until the destination is reached or the message expired. This method does not require every node to know the ranking of all other nodes in the system, but just to be able to compare ranking with the node encountered, and to push the message using a greedy approach. I call this algorithm BUBBLE-A, since each world/community is like a bubble. Figure 5.12 illustrates the BUBBLE algorithm and Algorithm 2 summarise the operations in a flat community (not hierarchical [7]) space.

This fits our intuition in terms of real life. First you try to forward the data via people more popular than you around you, and then bubble it up to well-known popular people in the society,

[7]We will discuss the hierarchical structures in the conclusion section.

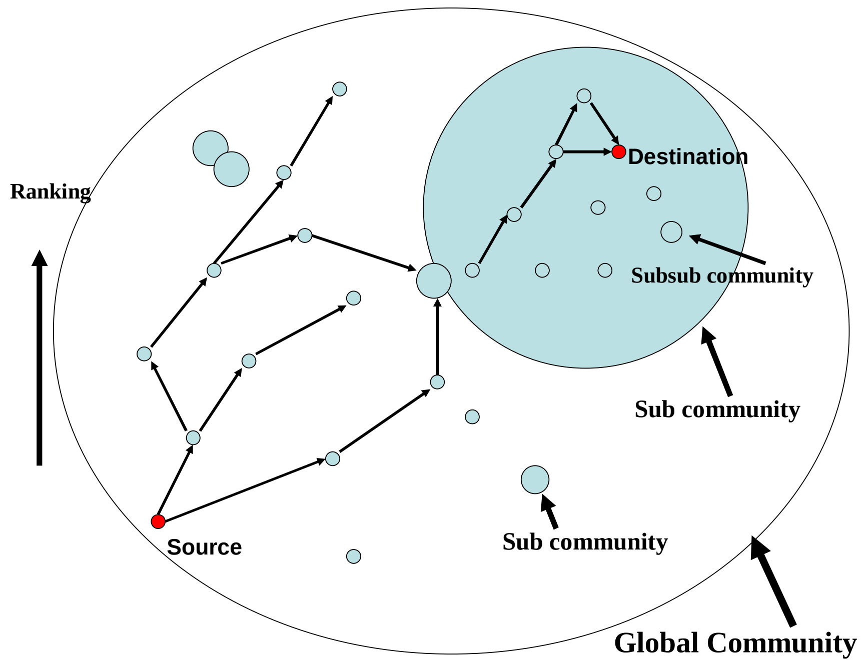


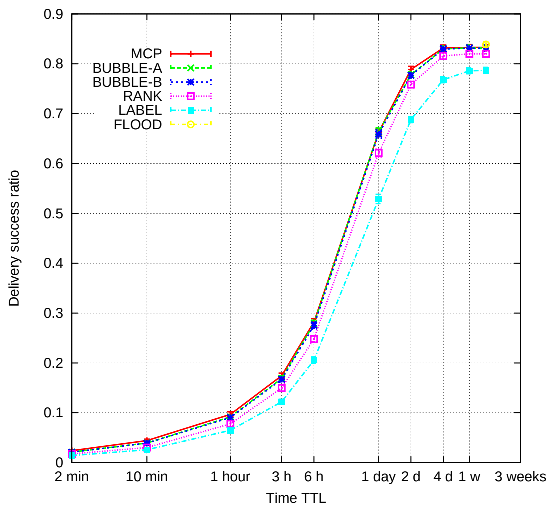
Figure 5.12: Illustration of the BUBBLE forwarding algorithm.

such as a postman. When the postman meets a member of the destination community, the message will be passed to that community. This community member will try to identify the more popular members within the community and bubble the message up again within the local hierarchy until the message reach a very popular member, or the destination itself, or the message expires.

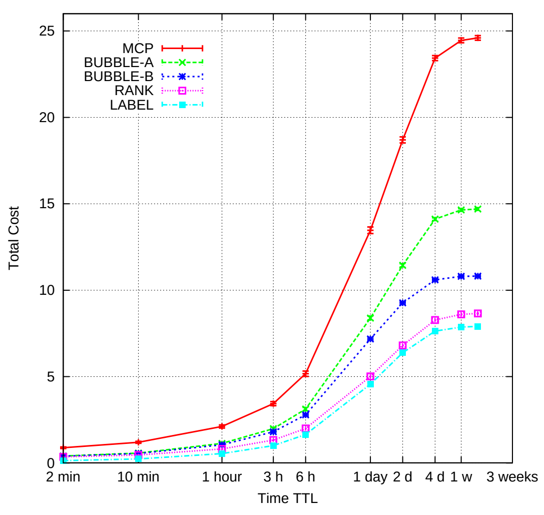
A modified version of this strategy is that whenever a message is delivered to the community, the original carrier can delete this message from its buffer to prevent it from further dissemination. This assumes that the community member would be able to deliver this message. I call this protocol with deletion, strategy BUBBLE-B.

We can see from Figure 5.13(a) that both BUBBLE-A and BUBBLE-B achieve almost the same delivery success rate as the 4-copy-4-hop MCP. [8] Although BUBBLE-B has the message deletion mechanism, it achieves exactly the same delivery as BUBBLE-A. From Figure 5.13(b), we can see that BUBBLE-A only has 60% the cost of MCP and BUBBLE-B is even better, with only 45% the cost of MCP. Both have almost the same delivery success as MCP.

[8]Here, I use the "perfect" centrality calculated using knowledge of the entire period, then use this knowledge in forwarding decisions throughout the same entire period. It seems a little bit unfair for the evaluations, but I will show in Section 5.7.2 that the centrality measured in the past is useful as a predictor for the future, hence it is a fair way to use "perfect" centrality. I have to take this approach because some of the datasets are really short in experimental periods and I cannot afford using a portion of a dataset to train the individual centrality.



(a) Delivery



(b) Cost

Figure 5.13: Comparisons of several algorithms on *Cambridge* dataset, delivery and cost.

5.6.2 Multiple-community Cases

To study the multiple-community cases, I use the *Reality* dataset. To evaluate the forwarding algorithm, I extract a 3-week session during term time from the whole 9-month dataset. Emulations are run over this dataset with uniformly generated traffic.

There is a total of 8 groups within the whole dataset. Figure 5.14 shows the node centrality in 4 groups, from small-size to medium-size and large-size group. We can see that within each group, almost every node has different centrality.

In order to make my study easier, I first isolate the largest group in Figure 5.14, consisting of 16 nodes. In this case, all the nodes in the system create traffic for members of this group. We can see from Figure 5.15(a) that BUBBLE-A and BUBBLE-B perform very similarly to MCP most of the time in the single group case, and even outperform MCP when the time TTL is set to be larger than 1 week. From Figure 5.15(b), we can see that BUBBLE-A only has 70% and BUBBLE-B only 55% of the cost of MCP. We can say that the BUBBLE algorithms are much more cost effective than MCP, with high delivery ratio and low delivery cost.

After the single group case, I start looking at the case of every group creating traffic for other groups, but not for its own members. I want to find the upper cost bound for the BUBBLE algorithm, so I do not consider local ranking; messages can now be sent to all members in the group. This is exactly a combination of direct LABEL and greedy RANK, using greedy RANK to move the messages away from the source group. I do not implement the mechanism to remove the original message after it has been delivered to the group member, so the cost here will represent an upper bound for the BUBBLE algorithms.

From Figure 5.16(a) and Figure 5.16(b), we can see that of course flooding achieves the best performance for delivery ratio, but the cost is 2.5 times that of MCP, and 5 times that of BUB-

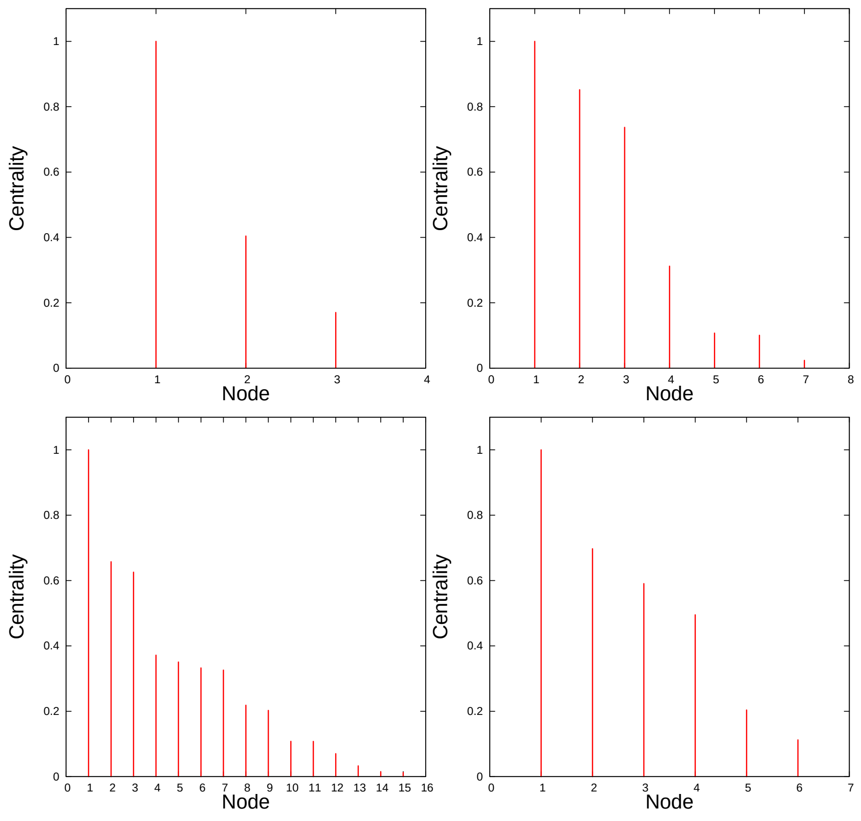
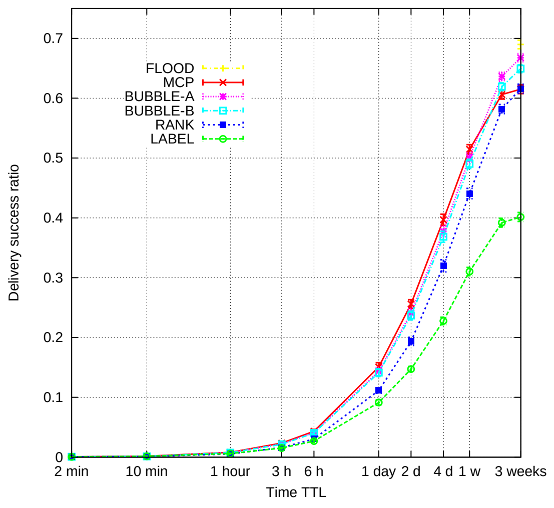
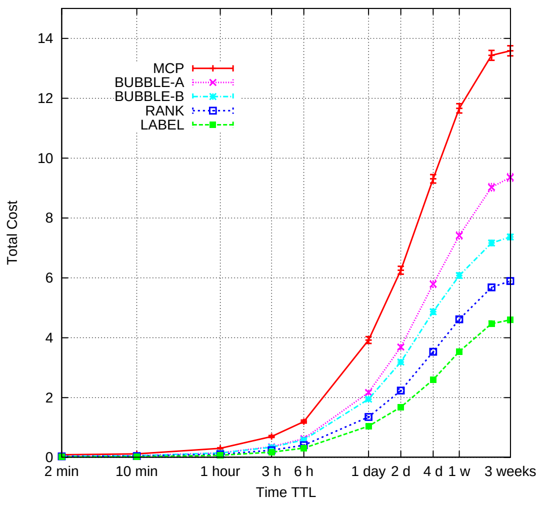


Figure 5.14: Node centrality in several individual groups (*Reality*).



(a) Delivery



(b) Cost

Figure 5.15: Comparisons of several algorithms on *Reality* dataset, single group.

BLE. BUBBLE is very close in performance to MCP in multiple groups case as well, and even outperforms it when the time TTL of the messages is allowed to be larger than 2 weeks.[9] However, the cost is only 50% that of MCP.

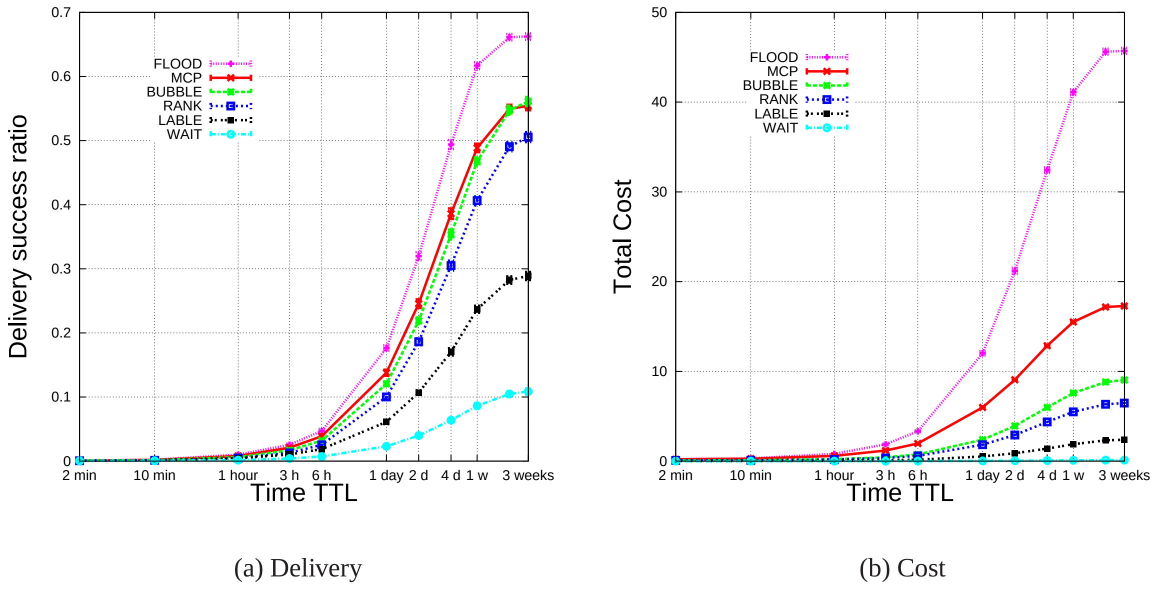


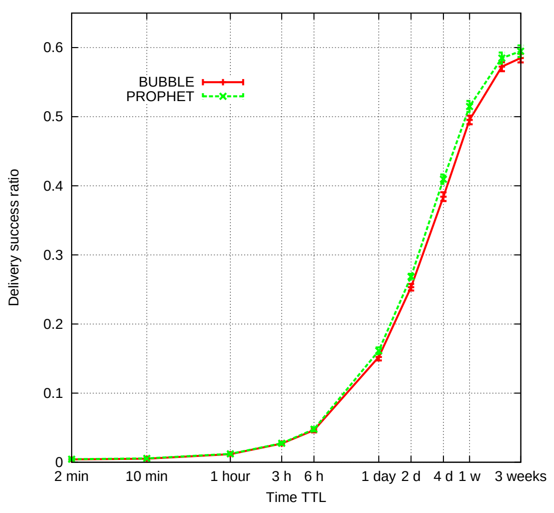
Figure 5.16: Comparisons of several algorithms on *Reality* dataset, all groups.

In order to further justify the significance of social based forwarding, we also compare BUBBLE with a benchmark 'non-oblivious' forwarding algorithm, PROPHET[LDS04]. PROPHET uses the history of encounters and transitivity to calculate the probability that a node can deliver a message to a particular destination. Since it has been evaluated against other algorithms before and has the same contact-based nature as BUBBLE (i.e. do not need location information), it is a good target to compare with BUBBLE.

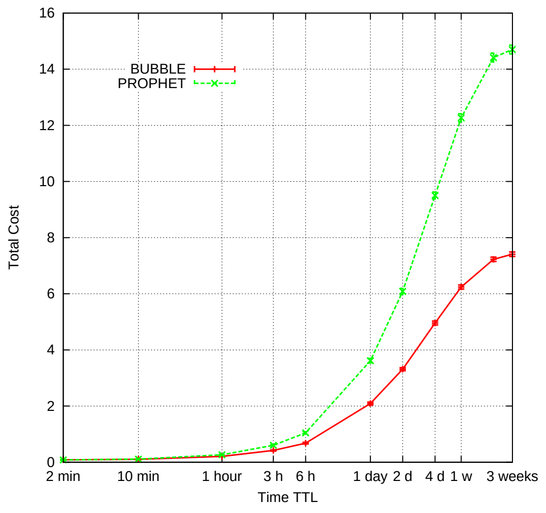
PROPHET has four parameters. We use the default PROPHET parameters as recommended in [LDS04]. However, one parameter that should be noted is the time elapsed unit used to age the contact probabilities. The appropriate time unit used differs depending on the application and the expected delays in the network. Here, we age the contact probabilities at every new contact. In a real application, this would be a more practical approach since we do not want to continuously run a thread to monitor each node entry in the table and age them separately at different time.

Figures 5.17(a) and 5.17(b) show the comparison of the delivery ratio and delivery cost of BUBBLE and PROPHET. Here, for the delivery cost, I only count the number of copies created in the system for each message as I have done before for the comparison with the "oblivious" algorithms. I do not count the control traffic created by PROPHET for exchanging routing table

[9] Two weeks seems to be very long, but as I have mentioned before, the *Reality* network is very sparse. I choose it mainly because it has long experimental period and hence more reliable community structures can be inferred. The evaluations here can serve as a proof of concept of the BUBBLE algorithm, although the delays are large in this dataset.



(a) Delivery



(b) Cost

Figure 5.17: Comparisons of BUBBLE and PROPHET on *Reality* dataset.

during each encounter, which can be huge if the system is large (PROPHET uses flat addressing for each node and its routing table contains entry for each known node). We can see that most of the time, BUBBLE achieves a similar delivery ratio to PROPHET, but with only half of the cost. Considering that BUBBLE does not need to keep and update an routing table for each node pairs, the improvement is significant.

Similarly significant improvements by using BUBBLE are also observed in other datasets. These demonstrate the generality of the BUBBLE algorithm, but for lucidity of the dissertation, I do not include the results here. There are several other 'non-oblivious' forwarding algorithms, for example the pattern based Mobyspace Routing [LFC06] by Leguay *et al.*, the location-based routing [Leb05] by Lebrun *et al.*, and adaptive routing by Musolesi *et al.* [MHM05]. Mobyspace and location-based routing both required location related information, but BUBBLE only use contact-based information. They are in different application categories and hence are not comparable. Adaptive routing provides a nice framework for choosing relays based on utility calculation, but strictly speaking it is not a forwarding algorithm. There is a social based forwarding algorithm called SimBet routing [DH07] proposed independently at the same time as BUBBLE, because they happened at the same time so I also did not do the comparison. SimBet routing uses ego-centric centrality but without community information to forward data. The RANK algorithm I introduced in this thesis would provide a delivery ratio upper bound for this algorithm. But overall, I evaluated BUBBLE against WAIT, FLOOD, the optimized MCP, LABEL, RANK, and the benchmark PROPHET, it is enough to justify the powerful performance of BUBBLE.

5.7 Making Centrality Practical

For practical applications, I want to look further into how BUBBLE can be implemented in a distributed way. To achieve this, each device should be able to detect its own community and calculate its centrality values. In Chapter 3 (also in [HYyCC07]), I have proposed three algorithms, named SIMPLE, K-CLIQUE and MODULARITY, for distributed community detection, and I have proved that the detecting accuracy can be up to 85% of the centralised K-CLIQUE algorithm. The next step is to ask how can each node know its own centrality in a decentralised way, and how well past centrality can predict the future.

The final contribution of this chapter is to provide early answers to these two questions.

5.7.1 Approximating Centrality

I found that the total degree (unique nodes) seen by a node throughout the experiment period is not a good approximation for node centrality. Instead the degree per unit time (for example the number of unique nodes seen per 6 hours) and the node centrality have a high correlation value. We can see from Figure 5.18 that some nodes with a very high total degree are still not

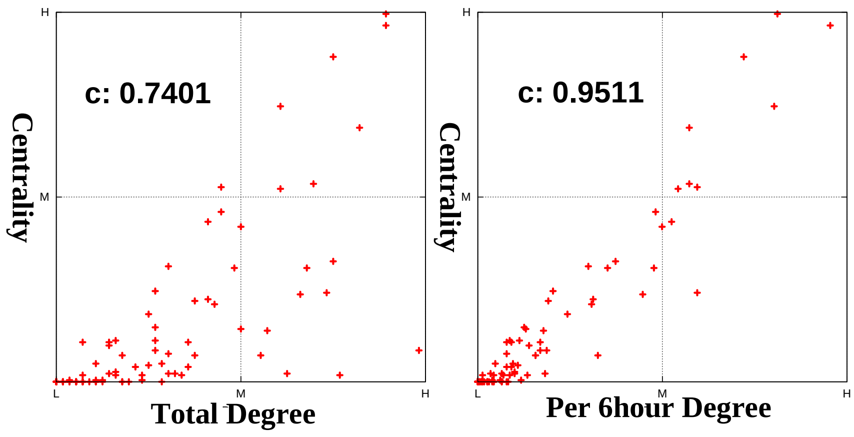
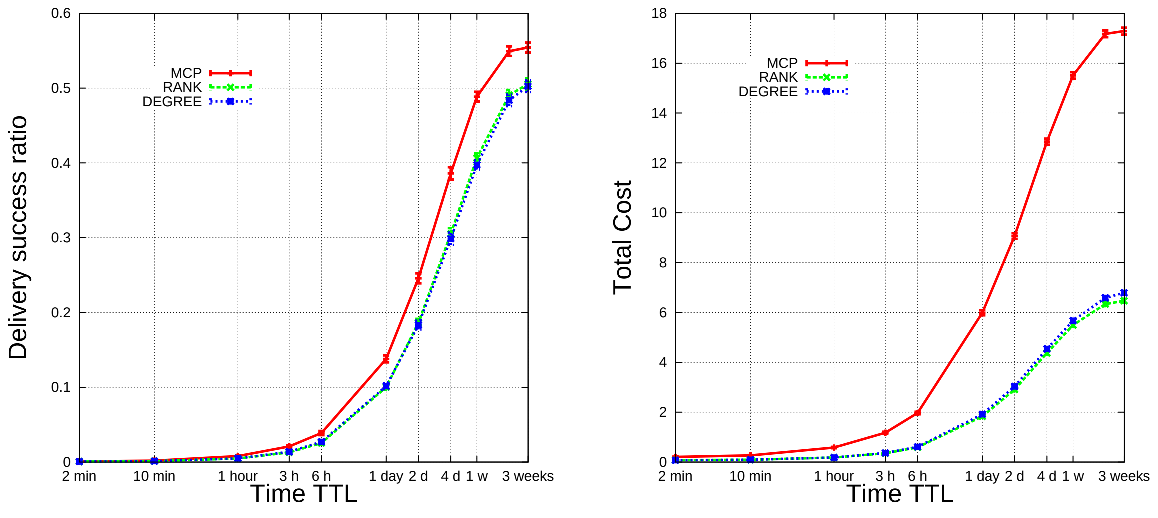


Figure 5.18: Correlation of rank with total degree and rank with unit time degree (*Reality*).

good carriers. It also shows that the per 6 hour degree is quite well correlated to the centrality value, with correlation coefficient as high as 0.9511. That means how many people you know does not matter too much, but how frequently you interact with these people does matter.

In order to verify that the average unit-time degree is as good as or close to RANK, I run another sets of emulations using greedy average unit-time degree (or I simply call it DEGREE) instead of the pre-calculated centrality. Figure 5.19(a) and 5.19(b) compare the delivery ratio and delivery cost of using greedy RANK and greedy DEGREE. We can see that RANK and DEGREE perform almost the same with the delivery and cost lines overlapping each other. They not only have similar delivery but also similar cost.



(a) Delivery (b) Cost

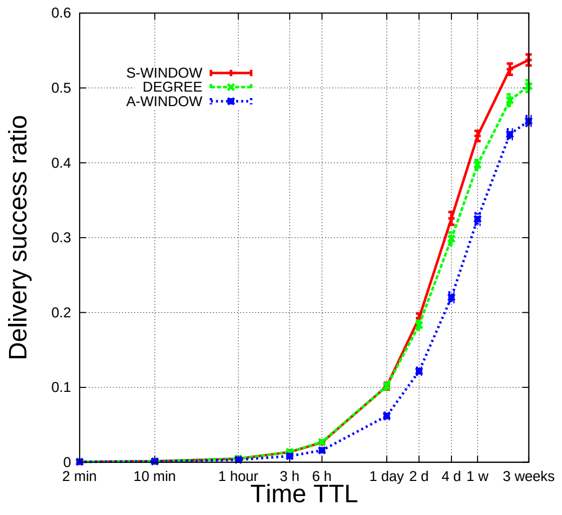
Figure 5.19: Comparisons of delivery ratio and cost of RANK and DEGREE on *Reality* dataset.

However, the average unit-time degree calculated throughout the whole experimental period is still difficult for each node to calculate individually. I then consider the degree for the previous unit-time slot (I call this the slot window) such that when two nodes meet each other, they compare how many unique nodes they have met in the previous unit-time slot (e.g. 6 hours). I call this approach single window (S-Window). Another approach is to calculate the average value on all previous windows, such as from yesterday to now, then calculate the average degree for every 6 hours. I call this approach cumulative window (C-Window). This technique is similar to a statistics technique called exponential smoothing [Win60] and I would like to do further theoretical investigation.

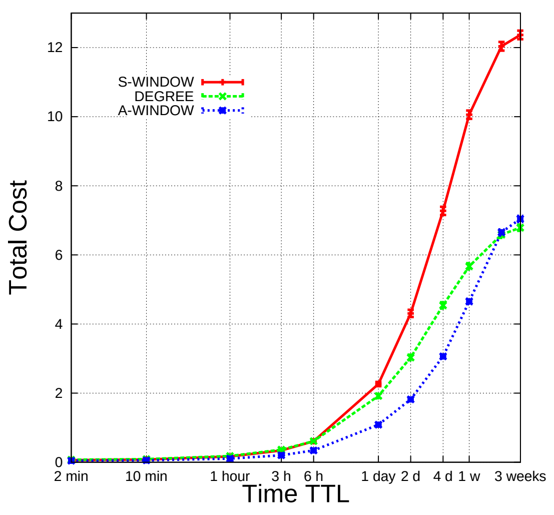
We can see from Figure 5.20(a) and 5.20(b) that the S-Window approach reflects more recent context and achieves a maximum of 4% improvement in delivery ratio over DEGREE, but at double the cost. The C-Window approach measures more of the cumulative effect, and gives more stable statistics about the average activeness of a node. However, its cumulative measurement is not as good an estimate as DEGREE, which averages throughout the whole experimental period. It does not achieve as good delivery as DEGREE (not more than 10% less in term of delivery), but it also has lower cost.

5.7.2 Human Predictability

The second question above can be generalised to: how much can human interaction be predicted from past contact history? In this section, I use vertex similarity, which has been well studied in citation networks, to study the predictability of human interaction from the contact graph. Additionally, I run emulations on traces to see how much past centrality can predict future centrality.



(a) Delivery



(b) Cost

Figure 5.20: Comparisons of delivery ratio and cost of DEGREE, S-Window and C-Window on *Reality* dataset

Vertex Similarity

There are several ways to compare structural vertex similarity in previous work. Two vertices are considered *structurally equivalent* if they share many of the same network neighbors,

$$\sigma_{\text{Jaccard}} = \frac{|\Gamma_i \bigcap \Gamma_j|}{|\Gamma_i \bigcup \Gamma_j|} \tag{5.1}$$

$$\sigma_{\text{cosine}} = \frac{|\Gamma_i \bigcap \Gamma_j|}{\sqrt{|\Gamma_i||\Gamma_j|}} \tag{5.2}$$

$$\sigma_{\text{min}} = \frac{|\Gamma_i \bigcap \Gamma_j|}{\min\left(|\Gamma_i||\Gamma_j|\right)} \tag{5.3}$$

where Γ_i is the neighborhood of vertex *i* in a network, which is the set of vertices connected to vertex *i* via an edge. $|\Gamma_i|$ is the cardinality of the set Γ_i, that is equal to the degree of the vertex *i*. The Jaccard index [Jac01] above is the same one introduced in Section 3.6, and the cosine similarity has a long history of study on citation networks [Sal89]. Here I use vertex similarity to measure the predictability of human interaction: we can compare the vertex similarity of the contact graphs over two days and tell how similar human interaction is on these two days. Averaging over all the vertices, we get an estimate for the whole population. I call this simply *graph similarity*. I have studied all three metrics, but the trends are similar, and so I just present the results of the classic Jaccard measurement here.

I look at the dataset of the *Reality* experiment from 1st February to 30th April 2005. The reason for choosing this period is that it is far from the new academic year so the human relationships are already relatively stable and also it is term time so the participants will be more active in the campus. I study the vertex similarity and the simple graph similarity for every two consecutive days and also for every pair of days against the date of the 1st of February for these three months. I consider it as a binary graph; I do not consider the weight for the edges, but just consider the existence of an edge. The three metrics proposed above do not apply to a weighted graph.

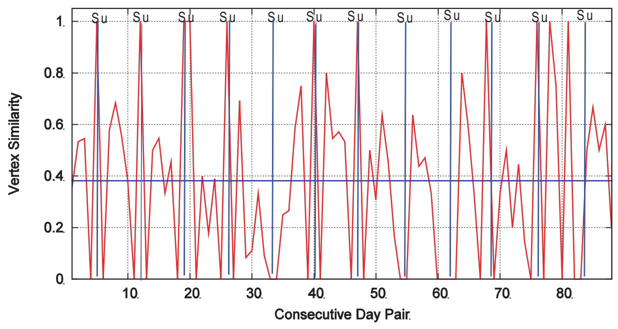


Figure 5.21: Vertex similarity of every consecutive day pairs of a single node

Figure 5.21 shows the Jaccard vertex similarity of an active node, i.e. a node with high centrality value, for the 88 consecutive day pairs. The horizontal line at the middle shows the average value. In the calculation, when two comparing vertices have both cardinalities equal to 0, I count their similarity to be 1, the maximum Jaccard similarity. We can see that the trough (minimum) points are corresponding to a change from weekday to weekend and also weekend to weekday; and the peak (maximum) points are corresponding to a transition from Saturday to Sunday, so there is always a peak surrounded by two troughs. We see that the nodes met by this node during the week-days are very different from the those nodes met during the weekend. For the weekend, the nodes met have a very high probability to meet again the second weekend day. But even during week-day, there are around 50% of the nodes met one day that will be met again the second day. This is the case for the active nodes, but for the less active nodes, i.e. the nodes with a low centrality value, they have the highest vertex similarity value: 1 almost everyday. These nodes usually see exactly the same nodes everyday. This also explain why they have low centrality values.

Figure 5.22 shows the simple graph similarity for the contact graphs of every consecutive day. We can see that the average value is as high as 0.7, for the whole population studied the human interaction pattern is quite predictable for every two consecutive days. The peaks here are also corresponding to the transition from a Saturday to a Sunday.

In order to see more clearly the phase transition from weekday to weekend, and also to look at whether there is any long-term attenuation for the human interaction in this system, I compare every day with the first day of the period I studied, which is 1st February and is a weekday. Figure 5.23 shows the vertex similarity of every day pair. We can see that the vertex similarity drops to zero from a weekday to a weekend transition and stays zero for the whole weekend. And we did not observe the long term attenuation effect from the graphs produced. Similar trends of changes are also observed in the graph similarity graph.

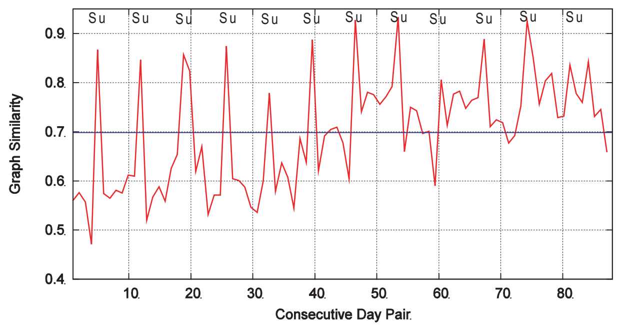


Figure 5.22: Simple graph similarity of every consecutive day pairs

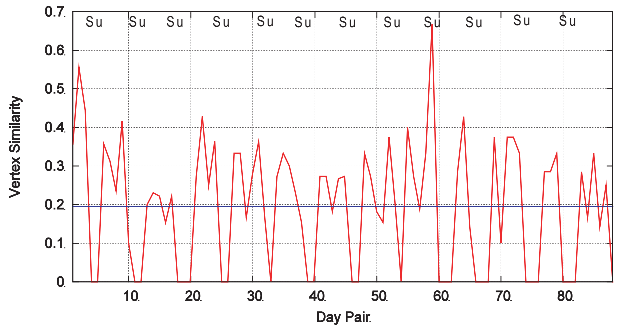


Figure 5.23: Vertex similarity of every day pairs with a randomly chosen weekday of a single node

But if we want to further look at whether the same node pair stay similar amount of time together for a day pair and also whether they meet for similar number of times everyday, we need to consider a weighted version of measurement for this kind of similarity. Since I cannot find useful metrics from the literature, I need to devise my own:

$$\sigma_{\text{weight}} = \frac{\sum_0^n \min(w_{it})}{\sum_0^n \max(w_{it})} \tag{5.4}$$

where $n = |\Gamma_i \bigcup \Gamma_j|$, $\min(w_{it})$ is the minimum and $\max(w_{it})$ of the weight for an edge connecting node *i* and one of its neighbours, node *t*, in the two graphs. If there is no edge in the graph, I count its weight to be 0. Here I count the number of contacts as the weight and then calculate the vertex similarity for all nodes, as well as the graph similarity. Figure 5.24 shows the weighted vertex similarity for every consecutive day pair for the same node as shown before. We still observe the transition from weekday to weekend and vice-versa. The horizontal lines in the middle show the average. It is around 0.3. That is not very high because of the transition from weekday to weekend and weekend to weekday would produce two 0 values. However, if we look at the whole population in Figure 5.25, we can see that even the contact frequencies of two consecutive days are quite predictable, with an average of close to 0.7.

I will look at the similarity of different time durations, the impact of different periods of the day (i.e. the nodes seen during the day time should be different from the nodes during night time), and different data analysis techniques such as correlation and matrix analysis will be used. The

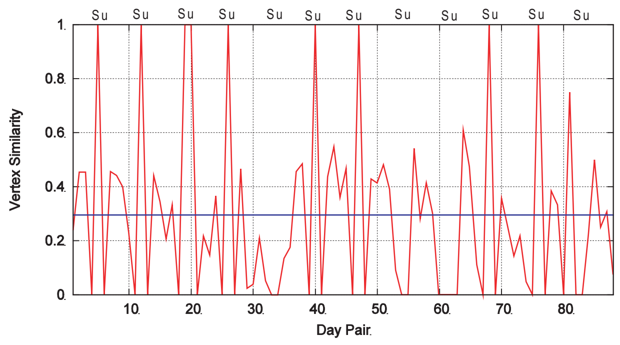


Figure 5.24: Weighted vertex similarity for every consecutive day pair of a single node

current result is limited to an academic campus but I will look at more complex environments in the future. An early conclusion I can make here is that daily human interaction is quite predictable. Nodes that met on one day have a high probability to meet again on the next day. This provides an indirect answer to the predictability of centrality as well.

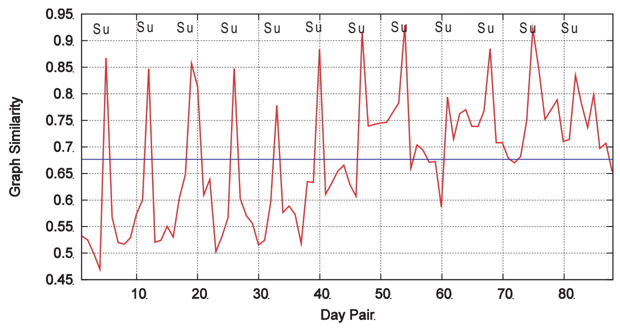


Figure 5.25: Vertex similarity of every day pairs with a randomly chosen weekday of a single node

Predictability of Centrality

In order to further verify whether the centrality measured in the past is useful as a predictor for the future, I extracted three temporally consecutive 3-week sessions from the *Reality* dataset and then run a set of greedy RANK emulations on the last two data sessions, but using the centrality values from first session.

Figure 5.26(a) and 5.26(b) show the delivery ratio and cost of RANK on the 2nd data session using the centrality values from the 1st data session. It seems that the performance of RANK is not far from MCP but with much lower cost, i.e. it is as good as running the emulation on the original dataset which the centrality values derived from. Similar performance is also observed in the 3rd data session. These results imply some level of predictability of human mobility, and show empirically that past contact information can be used in the future.

All these approaches, (DEGREE, S-Window, C-Window and predictability of human mobility) provide us with a decentralised way to approximate the centrality of nodes in the system,

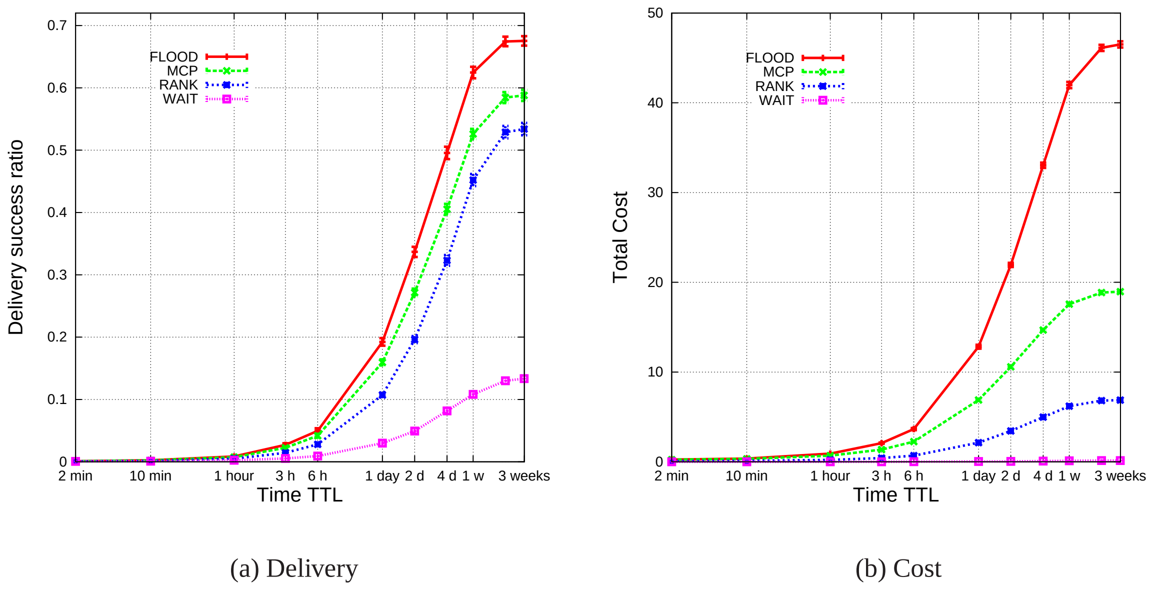


Figure 5.26: Delivery ratio (left) and cost (right) of RANK algorithm on 2nd data session, all groups (*Reality*)

and hence help us to design appropriate forwarding algorithms. Combining these approximate methods and the distributed community detection, we can put BUBBLE into reality. I will briefly discuss how distributed BUBBLE works for a city wide environment, but leave the evaluation details as future work when I can get a larger scale of dataset.

Suppose there is a network of mobile users, perhaps spanning an entire city, each device can detect its own local community using one of the three distributed detection algorithms (e.g. K-CLIQUE) from Chapter 3. At the same time, it also counts its own 6-hour-averaged degree (i.e. C-Window). Its global ranking can be approximated as its 6-hour-averaged degree for all nodes and its local ranking can be approximated as its 6-hour-averaged degree only for nodes inside its community. With all these metrics, each node can forward messages using BUBBLE.

5.8 Related Work

For distributed search for nodes and content in power-law networks, Sarshar *et al.* [SOR04] proposed using a probabilistic broadcast approach: sending out a query message to an edge with probability just above the bond[10] percolation threshold of the network. They show that if each node caches its directory via a short random walk, then the total number of accessible contents exhibits a first-order phase transition, ensuring very high hit rates just above the percolation threshold.

For routing and forwarding in DTNs and mobile ad hoc networks, there is much existing literature. Vahdat *et al.* proposed epidemic routing, which is similar to the "oblivious" flood-

[10]A percolation which considers the lattice edges as the relevant entities.

ing scheme I evaluated in this chapter [VB00]. Spray and Wait is another “oblivious” flooding scheme but with a self-limited number of copies [SPR05]. Grossglauser *et al.* proposed the two-hop relay schemes to improve the capacity of dense ad hoc networks [GT02]. Many approaches calculate the probability of delivery to the destination node, where the metrics are derived from the history of node contacts, spatial information and so forth. The pattern-based Mobyspace Routing by Leguay *et al.* [LFC06], location-based routing by Lebrun *et al.* [Leb05], context-based forwarding by Musolesi *et al.* [MHM05] and PROPHET Routing [LDS04] fall into this category. PROPHET uses past encounters to predict the probability of future encounters. The transitive nature of encounters is exploited, where indirectly encountering the destination node is evaluated. Message Ferry by Zhao *et al.* [ZAZ04] takes a different approach by controlling the movement of each node.

Recent attempts to uncover a hidden stable network structure in DTNs such as social networks have been emerged. For example, SimBet Routing [DH07] uses ego-centric centrality and its social similarity. Messages are forwarded towards the node with higher centrality to increase the possibility of finding the potential carrier to the final destination. In Chapter 4 (also in [HC07]), I use small labels to help forwarding in PSNs based on the simple intuition that people belonging to the same community are likely to meet frequently, and thus act as suitable forwarders for messages destined for members of the same community. The evaluation demonstrates that even such a basic approach results in a significant reduction in routing overheads. RANK algorithm introduced in this chapter uses betweenness centrality in a similar manner to SimBet routing. On the other hand, BUBBLE exploits further community structures and combines it with RANK for further improvement of forwarding algorithms. The mobility-assisted Island Hopping forwarding [NSDG06] uses network partitions that arise due to the distribution of nodes in space. Their clustering approach is based on the significant locations for the nodes and not for clustering nodes themselves. Clustering nodes is a complex task to understand the network structure for aid of forwarding.

Finally, I emphasise that I take an experimental rather than theoretical approach, which makes a further difference from the other work described above.

5.9 Conclusion and Future Work

Based on a diverse set of real world traces, I have detected characteristic properties of social grouping, and also showed how such characteristics can be effectively used in designing forwarding algorithms. I proposed the novel BUBBLE algorithm which is based on a delay-tolerant network environment, and built out of human-carried devices. It has similar delivery ratio to, but much lower resource utilisation than flooding, controlled flooding (e.g., MCP), and PROPHET, and hence is a powerful forwarding strategy for PSNs.

In Section 5.7.1 I chose a window size of 6 hours from the intuition that daily life is divided into 4 main periods, morning, afternoon, evening and night, each almost 6 hours. This appears to work, however, future work will look at how sensitive the system is to the choice of this period.

For betweeness centrality, I will look at the egocentric centrality which is a localised centrality measurement used in social network analysis and found to be correlated to the global social centrality [Mar02]. Using this egocentric centrality measure, we may not need concerning the global centrality measurement and we can make the problem more distributed.

On the forwarding aspect, I want to look at the use of real-world geographic landmarks as a part of the landscape within which forwarding algorithms can operate and be optimised further. In principle, BUBBLE is supposed to work with a hierarchical community structure, but because of the limited size of data (each experiment is not large enough for me to extract hierarchical structure), the current algorithm and evaluation focus on a flat community structure. This can later be extended to a hierarchical structure. I will further verify my results when more mobility traces are available.

Further experimental work involving large-scale experiments is required to confirm my results with more confidence in a wider variety of settings. Furthermore, I believe that it should be possible to abstract mathematical models of mobility that match my empirical results. We can use these models to generate further datasets with which to evaluate my and other forwarding systems.

I believe that this chapter represents a first step in combining rich multi-level information about social structures and interactions to drive novel and effective means for disseminating data in DTNs. A great deal of future research can follow. Based on the knowledge gained from this chapter and previous chapters, I will look at many-to-many communication as ongoing work in the next chapter.

Chapter 6

Conclusion

In this dissertation I have studied human mobility and interaction patterns by deploying experiments, inferred human communities from the traces, and introduced several social-aware forwarding algorithms for PSNs. Considering real deployment, several distributed community detection algorithms have also been suggested for mobile devices. In this chapter I summarise my contributions and describe some ongoing work and potential avenues for future research.

6.1 Ongoing Work

In the last chapter I introduced the BUBBLE algorithm which uses both community and centrality information to disseminate messages to some known destinations. This is a useful unicasting protocol for other communication paradigms to make use of. Unlike a traditional IP network, one-to-one is not necessarily the most popular communication mode for PSNs; instead multipoint communication such as data sharing and emergency announcement are more likely to be the killer applications, [1] because of the broadcast nature of the wireless channel and also given the intermittent connectivity of the network. I am extending this thesis work by looking at designing a social-aware overlay for publish/subscribe communication, which makes use of the social knowledge (community and centrality) I discussed in this thesis. Overlay nodes are the high centrality nodes in communities, for example a *closeness centrality* node has the best visibility to the other nodes in the community. Distributed community detection operates when nodes (i.e. devices) are in contact and subscription propagation is performed along with this operation. I validate my message forwarding algorithms for publish/subscribe with the mobility traces that I presented in Chapter 2.

Figure 6.1 depicts a publish/subscribe broker overlay, which is dynamically constructed through

[1] Although the expected delay for unicast messages is 1 day, the median delay can be much smaller as we have observed in the *Infocom06* dataset. For multi-point communication, the delay can be even more optimistic because of multiple sources.

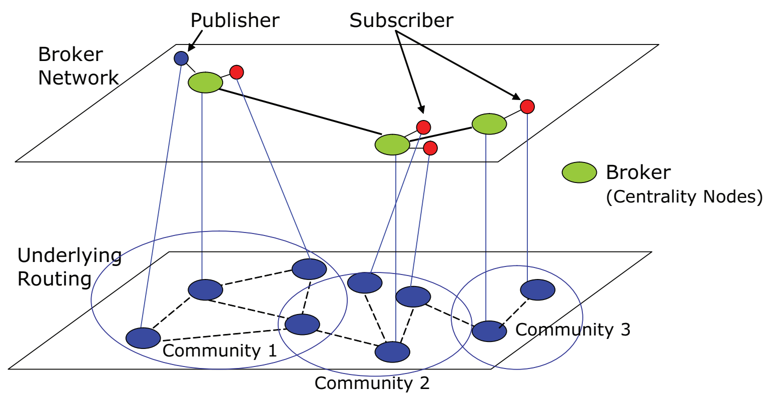


Figure 6.1: Overlay over Communities

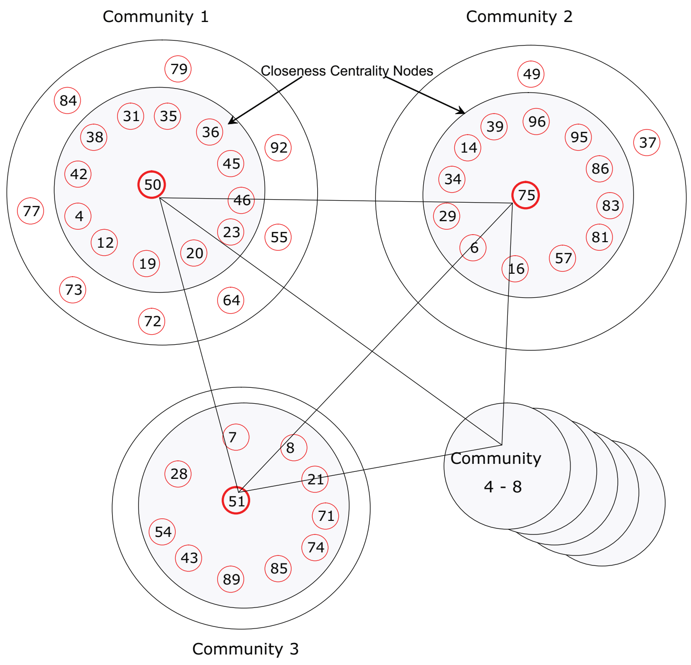


Figure 6.2: Community Structure

the gossipping stage for community detection. Construction of the broker overlay is independent from the underlying unicast routing algorithms.

Multiple centrality nodes can be used as a group of brokers. Fig. 6.2 depicts the community structure and closeness centralities detected in MIT Reality Mining trace. 8 communities are detected and the largest *Community 1* contains 21 members. *Communities 4-8* contain 3-4 members. There are 24 devices that do not belong to any communities, named *Loners*. Multiple centrality nodes are selected, which are in the inner circle. In *Community 1*, 13 nodes are closeness centrality nodes, thus, these nodes form a broker group. Alternatively one of the centrality nodes is named a single broker, which is marked at the centre of the circle in Fig. 6.2. All the detected centrality nodes have a single hop count to all the members of the community, and an average 93% of nodes in the community can take the role of event broker. Note that 24 nodes (25% of community members) do not belong to any communities or do not get a chance to be detected.

I have some early evaluation results of the overlay on the reality mining dataset (details can refer to our paper [YHCC07]), and I am now evaluating it on the other datasets and compare it with random gossiping. I am also looking at some potential killer PSN applications (i.e. hazard alert, data sharing, environment monitoring), which can make use of my overlay structure.

6.2 Future Research

Future research could include looking into more generic community detection algorithms, better modeling of temporal graphs, studying betweenness centrality on temporal graphs with different topologies, and large-scale mobility experiments. In the forwarding aspect, I also want to study multi-point communication, the impact of altruism on forwarding, the study of the impact of infrastructure at the city hot spots, strategies for deployment of infrastructure to improve communication, and comparing with other proposed forwarding algorithms.

As I mentioned in the above Ongoing work section, multi-point communication is an important research topic and I am looking into it, especially the issues of using community members to cache data in order to enable an ad hoc *Google* service.

The current distributed community detection algorithms have some inherent issues, such as reliance on choosing the appropriate threshold values as the algorithms' parameters, and the time-varying nature of some communities, which is a problem as the algorithms use accumulated information from the contact history, so obsolete community memberships would always persist. At present each device stores all its contacts without deleting. But in reality we need to consider *aging* for the contacts, since some previous contacts may become irrelevant after some time, but this takes up storage and may cause false-positives in the detection, especially for big networks.

For human communities we can further classify them into temporal communities and non-temporal communities, with temporal communities only valid for a particular time such as during a conference. According to the current distributed community detection algorithms, the temporal community will be outweighed by non-temporal contacts and will never be detected, but they can also be useful during that time period and we want to know about them. So it seems that in some situations it is desirable for a self-sustaining distributed community detection algorithm to have the ability to "forget" a dormant community; in other situations, it is desirable for the algorithm to be able to "remember" communities that are not currently active. The ideal solution is for each device to keep the records of the time and duration, for each encounter, so that it would be possible to compute on demand the local communities information within any given period. However, such computation would also require the histories of all its neighboring vertices (it is not enough to just know one's familiar set, but its familiar set's familiar set as well). Given the likely applications of distributed community detection algorithms are in environments with limited resources and bandwidth, it is not likely that the requirements of keeping copies of complete histories and the need to constantly exchange and update neighboring devices' histories can be satisfied. Therefore, a possible solution is the use of timestamping on the most recent encounters as a resource/accuracy compromise. I want to address these weaknesses in the future and also look at the issues about the temporal communities.

I have briefly introduced a simple method to model human interaction as a temporal graph with a random topology but a higher power-law coefficient for inter-community edges and lower power-law coefficient for intra-community edges (see Chapter 4). I want to improve the model by considering also the contact time distribution and graph with different topologies such as scale-free (with cut-off), small world, and real world social networks (e.g. Orkut[2], YouTube[3], and Flickr[4]) [MMG$^+$07].

I intend to correlate a node's centrality value to its degree of connectivity. This is probably true for a static graph but not necessarily for a temporal graph. As I found when studying the node betweeness centrality on the traces in Chapter 5, the number of people you know does not matter but how often you interact with them matters. I want to do a complete study on node centrality on temporal graphs with different topologies. I also want to look at the centrality from the social aspect, such as how the socio-centric centrality in a social network correlates to the temporal graph centrality, and also how the local measure, egocentric centrality, can be used for decentralised centrality approximation (definitions of socio-centric and egocentric centrality can be found in [Mar02]).

Altruism refers to behavior which is costly to the organism performing the behavior but which benefits other organisms. In a Pocket Switched Network (PSN), helping others to carry and forward data is an altruistic behavior. Correlated interaction refers to the idea that an organism of a given type might be more likely to interact with another organism of a same type than with a randomly chosen member of the population. I assume there is more altruistic behavior within communites/groups and less altruistic behavior between communities/groups, and I want to study how this heterogenous altruistic behavior affects information dissemination for both single source to single destination, and single source to multiple destinations.

Looking at the city structure in order to decide where to put the Haggle access points to improve communication can be an interesting topic as well. The idea is to map the city (i.e. Cambridge, Hong Kong) into a complex network with junctions as nodes and the roads connecting two junctions as edges (in this case, I count the actual width of the road to be the weight of the edge and length of the road as the distance, not the topological hop length) [CLP06]. I would then run my WNA implementation (community detection, edge betweenness, node centrality) on it and use this to decide where to put the access points (i.e. the min-cut between two community structures). We can even do an iMote experiment in the city of Cambridge to verify whether the number of Bluetooth devices detected matches the centrality of the streets.

On the forwarding aspect, I want to look at the use of real-world geographic landmarks as a part

[2]www.orkut.com
[3]www.youtube.com
[4]www.flickr.com

of the landscape within which forwarding algorithms can operate and be optimised further. In principle, BUBBLE is supposed to work with a hierarchical community structure, but because of the limited size of data (each experiment is not large enough for me to extract hierarchical structure), the current algorithm and evaluation focus on a flat community structure. This can later be extended to a hierarchical structure. I will further verify my results when more mobility traces are available.

I am planning to do a larger scale mobility experiment with the number of participants ranging from five hundred to one thousand using mobile phones and iMotes. A large experiment is necessary to verify my observations and results to eliminate the biases from limited sampling.

6.3 Summary

In Chapter 1 I began by introducing some background information about Delay Tolerant Networks (DTNs), Pocket Switched Networks (PSNs), forwarding in PSNs, and social network analysis. I then presented my thesis, that adding social knowledge can improve the forwarding efficiency in PSNs. I stated the four main contributions for this thesis including iMote experiments, inferring communities from the data, distributed community detection, and social aware forwarding algorithms. After that I gave a road map of the thesis.

In Chapter 2 I reported the iMote experiments I have conducted and also analysed the inter-contact time distribution of each node pair for all datasets, which was found to follow a heavy-tailed distribution with a power-law coefficient smaller than 1 in the range of 10 minutes to 1 day. I also presented analytical results about the impact of the power-law coefficient on forwarding and the empirical results of “oblivious” forwarding.

In Chapter 3 I showed how to apply weighted network analysis (WNA) and the K-CLIQUE community detection algorithms to infer community structures from the traces, which was then used for the community based forwarding study in Chapter 6. I also presented three distributed community detection algorithms with different computational complexity and accuracy. These algorithms achieve quite high accuracy when compared with the centralised detection algorithms.

In Chapter 4 I presented a simple social-aware forwarding algorithm called LABEL which makes use of *a priori* group information. I evaluated it on the *Infocom06* dataset, which has *a priori* affiliation information from the design of the experiment. I found out that LABEL can improve the forwarding efficiency in terms of delivery ratio and cost compared to the “oblivious” controlled flooding schemes. I also presented some early results of the modeling of temporal graphs using community structure and power-law distribution for inter-contact time.

In Chapter 5 I presented the BUBBLE algorithm, which makes use of both community and centrality information. I extracted human heterogeneity information from the traces and proposed this algorithm based on the observations. Most of the current proposed forwarding algorithms are not based on real human mobility analysis and are usually evaluated on simple mobility models. BUBBLE is a practical forwarding algorithm that is completely derived from human mobility measurements. I evaluated this algorithm by running emulations on the traces and found the forwarding efficiency to be significantly improved.

Appendix A

Haggle: A Clean-slate Architecture for PSN

In this chapter, I concentrate on describing a clean-slate architecture for mobile devices. Current mobile computing applications are infrastructure-centric, due to the IP-based API that these applications are written around. This causes many frustrations for end users, whose needs might be easily met with local connectivity resources but whose applications do not support this (e.g. emailing someone sitting next to you when there is no wireless access point). I identify the general scenario faced by the users of Pocket Switched Networking (PSN), and discuss why the IP-based status quo does not cope well in this environment. I present a set of architectural principles for PSN, and the high-level design of Haggle. Haggle is an asynchronous data-centric network architecture which addresses the mobility problem by raising the API so that applications can provide the network with application level Data Objects (DO) with high-level metadata concerning DO identification, security and delivery to user-named endpoints.

This chapter is joint work with Jing Su, Dr. James Scott, my supervisor Prof. Jon Crowcroft, Dr. Christophe Diot, Dr. Eben Upton, Dr. Meng How Lim, Dr. Ashvin Goel, and Dr. Eyal de Lara. The texts and figures of this chapter are mainly extracted and summarised from a paper [SHCD06] presented in the IFIP WONS2006 conference with Dr. James Scott as the first author and me as a second author and also from another paper in Ubicomp 2007 with Jing Su as the first author and me as the third author. My main contribution is in the architecture design at the WONS paper stage and Jing Su's main contribution is in the prototyping and refining of the architecture at the Ubicomp paper stage. Dr. James Scott is the the manager for the architecture design and prototyping. The other collaborators also have strong input in both software engineering and technical opinions.

A.1 Introduction

Miniaturisation, Moore's Law and convergence have had a profound impact on portable devices, such as smart phones, notebooks, and PDAs. The result is that people are able to carry their previously desktop-based computing environments with them, with the aim of having ubiquitous access to applications such as email and web browsing in an always-on, always-available fashion. However, the low speed (e.g. GPRS) , high price (e.g. GPRS) and constrained availability (e.g. 802.11) of wireless Internet access means that these devices are often disconnected from the Internet, or have only a slow or expensive connection. These devices use the same, OSI-layered, IP based networking approach as desktop PCs, which assume a fixed network. This fixed network design sometimes performs badly or not at all in the environment that mobile devices find themselves in, which can bear more resemblance to Pocket Switched Networks [HCG$^+$05] or Delay Tolerant Networks [Fal03].

The Underlying Problem

In Section 1.3, I listed two motivating examples for PSNs. The root cause of the deficiencies highlighted in the stories lies in the current network architecture for mobile devices (the IP suite of protocols and the Berkeley sockets API), which presents applications with a synchronous, end-to-end connectivity model using numeric addresses for endpoints. In order to satisfy user-level tasks such as messaging and web-browsing, applications are effectively forced by this model to act in ways that include inherent reliance on networking infrastructure, i.e. Internet connectivity.

Due to the use of a synchronous model, applications are forced to become aware of the connectivity state of the node and to handle changes in this state, or (typically) simply assume always-on connectivity and avoid solving the problem. By employing end-to-end connections, applications are prevented from making use (without extensive and explicit support) of network routes that may involve non-contemporaneous connectivity. By requiring numeric addresses, user-memorable endpoint identifiers such as *user@domain.org* and *www.server.com* must be translated before the interface can be used, forcing a reliance on the presence of DNS.

In reality, while our devices may often have cheap, fast Internet connectivity for some periods (e.g. when we are at home or work), at other times they are disconnected from the Internet, or only connected through an expensive and/or slow network (e.g. GPRS, pay-to-use 802.11 APs). However, while they are disconnected, devices may often be connected to other devices in the neighborhood, and, as described in the motivating examples above, this limited connectivity may often be enough to provide significant value to users if it could be put to work.

Chapter Contributions and Structure

In the rest of this chapter, I present the Haggle architecture, a ground-up redesign of networking for mobile devices, to support the mobile user scenario.

The contributions of this chapter are as follows. First, I give an overview about PSN (Section A.2). Then I present the problem with the status quo (Section A.3). After an overview of the core concepts behind Haggle, (Section A.4), I present a detailed description of the Haggle architecture (Section A.5). While many of the ideas that are integrated into Haggle are built on existing research[1], a key contribution of this chapter is their organisation into a coherent architecture. Other contributions include the comparison of Haggle architecture and the current DTN architecture (Section A.6), and the description of the prototype and applications (Section A.7). Finally I conclude this Chapter in Section A.8.

A.2 Pocket Switched Networking

In designing a new network architecture, it is first important to define the scenario in which that architecture will be used. IP, for example, was designed against a backdrop of a multitude of existing networks, and with the primary needs being resilient end-to-end communications in the presence of node failures, as befits its originator, the US Department of Defense [Cla88].

PSN is the term we use to describe the situation faced by today's mobile information user. Such users have one or more devices, some/all of which may be with them at any time, and they move between locations as part of a normal schedule. In so moving, the users can spend some (or much) of their time in islands of connectivity, i.e. places where they have access to infrastructure such as 802.11 access points (APs) which they can use to communicate with other nodes via the Internet. They also occasionally move within wireless range of other devices (either stationary or carried by other users) and are able to exchange data directly with those devices.

Thus, in PSN, there are three methods by which data can be transferred, namely, neighborhood connectivity with local devices, infrastructure connectivity to the global Internet, and user mobility, which can physically carry data from place to place. For the former two methods, the connectivity is subject to a number of characteristics, including those of bandwidth, latency, congestion, synchronicity (e.g. email or SMS are asynchronous, while ad-hoc 802.11 is synchronous), the duration of the transfer opportunity (i.e. the time till the device moves out of range), and also monetary cost (usually only for infrastructure). For the latter method of user mobility, users acting as data mules can transfer significant amounts of data, and while users movements cannot in general be controlled, they can be measured, and patterns in those movements can be exploited. I will go into the details of mobility measurement in the next chapter.

[1]I provide references in the main body of the text rather than in a separate section.

In addition to the issue of network connectivity, we must also consider the usage model for PSN. While different applications have different network demands, we can distinguish particular broad classes which are known to be useful: (a) *one-to-many* where one node needs to transfer data to a user-defined destination. The destination may be another user (who may own many nodes), all users in a certain place, users with a certain role (e.g. police), etc. The key point is that, often, the destination is not a single node but is instead a set of nodes with some relationship, e.g. the set of nodes belonging to a message recipient. (b) *flooded query* in which a device requires data of some sort, e.g. the current news. The source for this data can be any node which is reachable using any of the three connectivity types, including via infrastructure (e.g. a news webpage), neighbors (e.g. a recent cache of a news webpage) or mobility (e.g. the arrival of a mobile node carrying suitable data). In both classes described above, the endpoints of a network operation are no longer described by network-layer addresses, but are instead a set of desirable properties. As a result, general network operations no longer have single source and destination nodes.

Finally, in PSN situations, resource management is a key issue. Mobile devices have limited resources in terms of storage, network bandwidth, processing power, memory, and battery. The latter is perhaps the most important, since the others can potentially be reclaimed without the users assistance, while charging the battery requires the user to perform the physical act of plugging it in, and restricts the devices' mobility while charging. Other resources are also precious, particularly in the face of demands imposed by the usage scenarios above, where devices may need to use storage and network bandwidth to help forward messages for other devices. However, there is also much cause for optimism: storage capacities are increasing exponentially, wireless networking has the useful property of spatial reuse, and processing power on mobile devices is growing with Moores Law. For power, many devices are plugged in more often than not, e.g. notebook computers, and low-power electronics allow current mobile phones to last for many days on a single charge.

From the discussion above, I extract three motivations for a networking architecture in the PSN environment, in order of importance:

- Allow applications to take advantage of all types of data transfer (neighborhood, infrastructure, mobility) without having to specifically code for each circumstance
- Allowing networking endpoints to be specified by userlevel naming schemes rather than node-specific network addresses, thus each network operation can potentially involve many endpoints.
- Allowing limited resources to be used efficiently by mobile devices, taking into account user-level priorities for tasks.

A.3 Problem with the Status Quo

Current applications perform badly in the PSN environment, since they are typically designed around some form of infrastructure which is not always available. While some applications can cope with infrastructure blackout, e.g. with a disconnected or offline mode, most do not. Direct, neighborhood connectivity is used by very few widely used applications, and human mobility is deliberately used by almost none. Thus, when infrastructure is not present, users are presented with huge inconveniences since the applications which are familiar to them stop working, and are forced to take on the task of understanding these situations so that they can be productive despite this application failure. For instance, users may require many alternative applications in order to do a single task depending on the situation, e.g. a file can be exchanged by email, by putting it on a website for download, by using an instant messaging client, by direct Bluetooth or infrared transfer. More likely, the user will simply invest in a USB key and manually bypass the huge inconvenience of the status quo.

The root cause of this is the fact that applications are provided with a networking interface that only understands streams of data directed at anonymous numeric endpoints (namely TCP/IP). This forces developers to implement protocols for naming, addressing and data formatting internally in the applications themselves, e.g. SMTP, IMAP and HTTP. While at the GUI level, applications have general user-level tasks such as “send this file to James”, once a particular network protocol such as SMTP is imposed on that task, it becomes a more specific task, e.g. “send this file to the server pointed at by the MX record in the DNS record of the domain name part of *james.w.scott@intel.com*”. The latter task is specific to a particular kind of connectivity scenario, in this case infrastructure-based. It is therefore impossible to execute even if James’ device is in the neighborhood at that time i.e. even if the user-level task could be satisfied.

Another problem with the current networking API is that it is synchronous. Applications cannot indicate a network task to be performed and then exit, since finished applications have all their TCP/IP sockets closed. For example, an email application with pending outgoing email in the outbox will not be able to use a passing AP to send this email if the application is not running when the AP is passed. Therefore, an application in the PSN environment has to be constantly on and monitoring the connectivity status of the device. This increases the complexity of a disconnection-aware application, since it must be able to wait through periods of bad connectivity and detect and perform networking actions when a suitable endpoint is again visible. It also increases the load on mobile device resources, since many applications would have to be present in the background at all times.

Another problem is that persistent user data is kept by applications in a file system which, in the current node architecture, is disconnected from the networking system. This means that all sharing of data between nodes must often be conducted by applications themselves. The biggest example of this is the device synchronisation problem. When a user has multiple devices, they must explicitly run an application on each which pulls their data out of the file system and shares it with their other device(s). Such synchronisation is often a source of much inconvenience for users, since the sync tools must understand the different ways that each user application uses the file system to store data and metadata, and often has to translate it so that different applications can be synced with the same data. Another example is in distributed web caching. The exact web page that a user wants may be in the cache of a neighboring node, but since web browsers do not explicitly support the transfer, there is no way to get this off the neighbour's file system and into the network to be shared with the user.

The final problem identified is that applications have no easy way to prioritise the use of a mobile device limited resources. These resources include persistent storage, network bandwidth, and battery energy. Currently, an application such as a web browser must estimate by itself how much of the storage can be used for non-critical history caching, or how much network bandwidth should be used for pre-fetching of web pages. This decision is often passed on to the user, who might have to adjust settings manually, at the application level (e.g. how much disk to use as cache), at the hardware level (e.g. turning on or off wireless network interfaces depending on the battery level), or by only running certain apps when they do not want to prioritise network bandwidth for other tasks (e.g. network-hungry file-sharing apps). These controls are coarse at best, and require expert understanding in order to properly exercise them. The result is that resources are often used inefficiently.

A.4 Core Concepts of Haggle

In previous work [SHCD06] I explored the principles behind the design of Haggle (though I had not built a working prototype at that time). In this dissertation, I reiterate the key concepts before diving into the architectural description that follows.

The key idea behind Haggle is to have a data-centric architecture [ASS$^+$] where applications do not have to concern themselves with the mechanisms of transporting data to the right place, since that is what has made them infrastructure-dependent. By delegating to Haggle the task of propagating data, applications can automatically take advantage of any connection opportunities that arise, both local neighborhood opportunities and connectivity with servers on the Internet when available. I identify four design decisions for Haggle that follow on from this.

A.4.1 Data Persists inside Haggle

The data on each node in Haggle must be visible to and searchable by other nodes (with appropriate security/access restrictions applied). This facilitates operation of our motivating web example, in that the public webpage needed by one person can be found despite it being in another person's device. In practice, this means that Haggle must manage persistent data storage for applications, instead of applications storing data in a separate file system.

A.4.2 Networking Protocols inside Haggle

Any application-layer networking protocol includes implied assumptions about the type of network available. For example, client-server protocols such as SMTP, POP and HTTP assume that Internet-based servers are contactable. With Haggle, I place networking protocol support inside Haggle itself, allowing me to present a data-centric rather than connection-centric abstraction to applications.

A.4.3 Name Graphs supporting Late Binding

Since Haggle aims to be infrastructure-independent, it must be able to use protocol-independent names for delivery (since many protocols imply infrastructure of one sort or another). Since we are in an environment where we cannot predict the best path for data a priori, we must perform late binding from protocol-independent names such as a person's name to protocol-specific names such as MAC addresses or email addresses [AWSBL]. Haggle therefore maintains its own naming repository (it obviously cannot rely on remote look-up of this data), with mappings from user-level names to protocol-specific names specifying the various ways to get to the user-level name. Furthermore, the whole set of mappings (the "name graph") is transmitted along with the data, allowing even intermediate (i.e. non-source) nodes to bind to protocol-specific names as late as possible.

A.4.4 Centralised Resource Management

One role of the networking architecture on every device is to decide what to do with each of its network interfaces *now*. In the current architecture, this decision does not take into account resource management — the decision to spend resources on something is taken by applications individually. This makes it very hard for applications to be proactive, since they must make sure themselves that only a suitable level of resources is used. Haggle therefore contains a centralised resource management component, which decides on a cost/benefit comparison basis what tasks it chooses to perform on each network interface at a given moment.

A.5 Haggle Architecture

The Haggle architecture is shown in Figure A.1. Haggle is at a macro-scale comprised of six *Managers*, the Data, Name, Forwarding, Protocol, Connectivity and Resource Managers. In addition, many of the Managers themselves have well-defined abstractions for their contents, as shown in italics/parentheses on the diagram - e.g. the Protocol Manager encapsulates a number of "Protocol" objects.

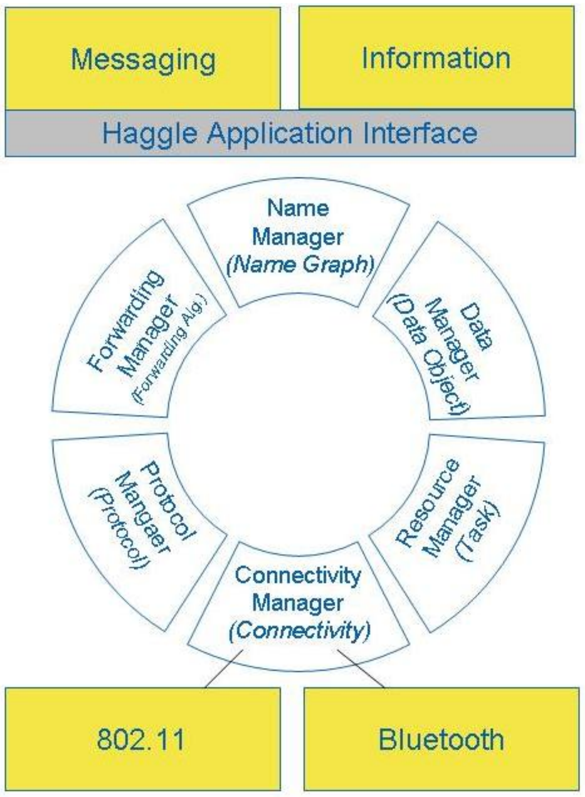
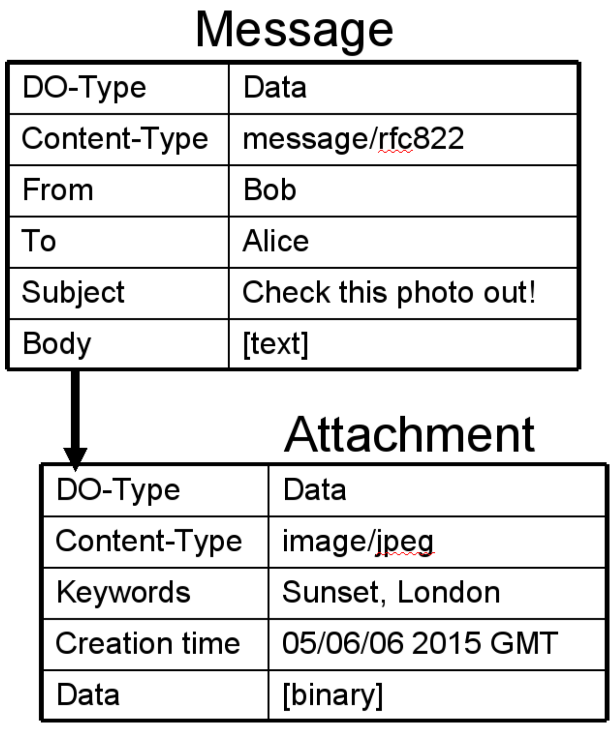

Messaging
Information
Haggle Application Interface
Name Manager (Name Graph)
Data Manager (Data Object)
Resource Manager (Task)
Connectivity Manager (Connectivity)
Protocol Mangaer (Protocol)
Forwarding Manager (Forwarding Alg.)
802.11
Bluetooth

Figure A.1: Haggle overview

Haggle is a layerless architecture, in that we do not pass data and control signals up and down between layers as for the current network architecture. Instead, all Managers provide interfaces which other Managers can communicate with. In terms of the current model, Haggle spans the link layer through to the application layer, inclusive. Link layer functionality in Haggle includes, for example, the choice of whether the 802.11 interface is in infrastructure mode or ad hoc mode, while Haggle's Protocols include application-layer protocols such as SMTP and HTTP. Rather than present a "cross-layer design" where layering is deliberately broken, I instead acknowledge that this model is not appropriate for Haggle. The key value of layering, in that between layers there are well-defined abstract interfaces facilitating modularity, is kept: the six Managers provide abstract interfaces and are modular in that they can be replaced independently.

As there is no top layer, the API that Haggle provides to applications is composed of a subset of the APIs that each Haggle manager provides to each other. In this chapter I do not list the APIs explicitly due to an excessive level of detail — interested readers are referred to *http://cvs.sourceforge.net/projects/haggle/* where these interfaces are available.

I now describe the design of each of the managers, including the key data abstractions and components, and how they communicate to perform networking tasks. In this section I restrict myself to describing architectural decisions, and do not discuss specific implementations (e.g. the SMTP protocol, or the 802.11 connectivity) — this is left to later in the chapter where I discuss the prototype that has been built and evaluated. At the end of this section, I discuss



Message

DO-Type	Data
Content-Type	message/rfc822
From	Bob
To	Alice
Subject	Check this photo out!
Body	[text]

Attachment

DO-Type	Data
Content-Type	image/jpeg
Keywords	Sunset, London
Creation time	05/06/06 2015 GMT
Data	[binary]

Figure A.2: Example DOs: Message and Attachment

potential security and privacy issues that Haggle raises.

A.5.1 Data Manager

As stated previously, Haggle maintains users' data persistently rather than relying on a separate file system. Haggle's data format is designed around the need to be *structured* and *searchable*. In other words, relationships between application data units (e.g. a webpage and its embedded images) should be representable in Haggle, and applications should be able to search both locally and remotely for data objects matching particular useful characteristics. I draw inspiration from desktop search products (e.g. Google Desktop) which have changed the way that many users file and access their data [CDT06], allowing us to avoid having to methodically place data in a file/directory structure. I propose that applications can use a combination of structured data and search, with the former providing the kind of capabilities expected of a traditional file/directory system, and the latter allowing applications to easily find and use data that they themselves did not store.

Data Objects

The data format is simple. A Data Object (DO) comprises many *attributes*, each of which is a pair consisting of a *type* and *value*. Types and values are typically strings, though some values may also be binary packed representations, e.g. the data in an mp3 file. I encourage and expect applications to expose as much *metadata* as possible about an item of application data using attributes, including application data. Two example DOs are shown in Figure A.2, representing a message from Bob to Alice, and a photo of the sunset. Note that Haggle does not require users to enter more metadata about their objects than applications would require themselves; the value of exposing metadata is in searchability using DO filters described later.

In order to facilitate multi-application environments and to avoid cache consistency issues, DO attributes are immutable after creation. (Haggle itself may mutate attributes for internal record-tracking, but applications may not). Applications must create a new DO instead of modifying a DO, and cause their existing links and claims to point at the new DO. This provides useful guarantees for applications that their data will not be modified "under their feet", although the disadvantage is that if they wish to use the most up-to-date version of data, they must be proactive, and use the search functionality described later to get updates either proactively or reactively. Another potential disadvantage of copying DOs, that of the time and storage costs of data replication, can be minimised by using standard copy-on-write techniques present in many filesystems.

Links between DOs

DOs can be linked into a directed graph. Links can take two forms. The first is to link data to embedded or prerequisite data, e.g. a photo album's metadata can link to the set of photos in the album, a webpage can link to its embedded objects, or (as shown in Figure A.2) an email can link to its attachments. This provides applications with a way to structure data, akin to the way that some applications use the placement of files in a common directory but more explicit. It allows Haggle to keep track of the prerequisite objects that must be shared alongside a top-level object in order to properly transmit a given application data unit. The second purpose of linking is for applications to themselves link to the DOs which they require for their operation, which can be regarded as an "ownership claim." In this way, many applications can claim the same DO, e.g. a photo gallery application can claim a photo that is linked to by a message (which brought it into the node) which is in turn claimed by the messaging application. Linking and claiming are accomplished using the same mechanism. I use the two terms to differentiate between the parent being another DO or a different entity.

Since Haggle allows many applications to claim DOs, it does not have a "delete" call, instead, just "unlink". When the last link is removed from a DO, it becomes eligible for garbage collection, though this is not necessarily performed immediately since the node may have plenty of persistent storage space. The data remains searchable even in its unclaimed state, which is an advantage since data is not lost unless space is actually required for new data.

DO Filters

In addition to the ability to retrieve DOs via a unique ID provided at creation time, the Data Manager also supports searching of DOs using a "DO filter". This comprises a set of regular-expression-like queries about the attributes of an object, e.g. “mime-type” EQUALS “text/html” AND “keywords” INCLUDES “news” AND “timestamp” $\geq$ (now() - 1 hour) would return DOs matching recent news webpages. A filter can be made persistent, and since it is itself stored in a DO, it can be sent remotely. This flexibility allows a single mechanism to be used for multiple purposes: a non-persistent local filter is a search on local data, a persistent local filter is a registration of interest in incoming data of a particular type (which functions analogously to a TCP socket “listen”), a non-persistent remote filter is a request for data which is sent across the network as appropriate (depending on the Forwarding Algorithms, Protocols and Neighbors available), and a persistent remote filter allows “subscriptions” to particular data to be registered with other nodes (e.g. a home PC registers interest in receiving all photos generated on a mobile phone).

The Data Manager is responsible for matching DO filters to DOs, and performs this whenever new data appears (which may match an existing persistent filter) or whenever a new filter appears (which may match existing DOs). If there are matches, then the source of the filter (whether local or remote) is notified. The ability of Haggle to unilaterally, without invoking application processes, answer remote queries is a key feature. It facilitates sharing of information between nodes beyond what we have today, since once the information is provided to Haggle, any and all connection opportunities that the node sees can result in the sharing of that information, given appropriate security concerns such as encryption and access control.

A.5.2 Name Manager using Name Graphs

Endpoint descriptions for data transmissions in Haggle are not performed in the usual method of the nested headers found on the front of current physical-layer packets (e.g. Ethernet address, IP address, TCP port, and SMTP’s RCPT TO field describing the endpoint for an email message). This is because I aim to be able to make use of any available ad-hoc or infrastructure connectivity opportunity. Since we cannot assume knowledge of the best end-to-end path, either when a communication is generated or even at an intermediate node once the communication is in-transit, we cannot perform the ahead-of-time directory lookups that are currently used to map a user-level endpoint, e.g. “Bob Smith”, to the SMTP’s RCPT TO field (b@a.org) and so on, in order to construct those addresses ahead of time. Not only is it the case that some of these lookups require infrastructure services such as DNS that may not be available, but even more importantly it is not possible to perform the initial name-to-email-address mapping that is implicit in the users’ choice of an email client rather than an Instant Messaging or mobile phone text message (SMS) client. The choice of client program by the user is currently equivalent to making them choose a networking protocol (e.g. email implies use of SMTP) and may be equivalent to making them choose a device on which the receiver will receive the message (e.g. text message implies use of a particular phone).

We require a more general form of naming notation that allows late-binding of user-level names, independent of the lower-level addressable name, as proposed in i3 [SAZ$^+$]. We achieve this

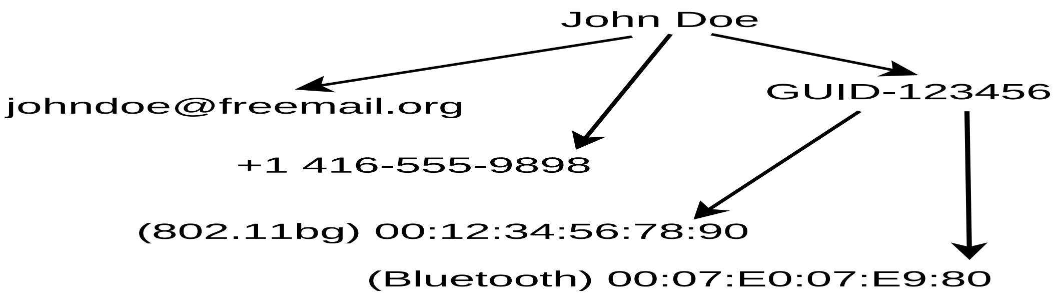


Figure A.3: Example of a name graph

by using *name graphs*, inspired by the intentional naming system (INS) [AWSBL], which are hierarchical descriptions of all known mappings from a user-level endpoint to lower-level names (which may imply particular protocols/connectivity methods), and by using the whole name graph as the "recipient" for a message, both on the source node and at intermediate nodes for the message. This is one of the "layerless" aspects of Haggle, and it contrasts with the existing Internet architecture where names are only meaningful at particular layers of the protocol stack.

What's in a Name?

An example name graph is shown in Figure A.3. These graphs span from top-level nodes such as personal names through to leaves comprising persistent methods of reaching them (e.g. an email address), but do not include transient addressing data for those methods (e.g. an IP address for the email server, or MAC address for the next-hop). Let us first discuss the choice of this partition [KKP].

In Haggle, we regard all of the nodes in a name graph as "names", and *any* of these names can be an "address" if there exists a suitable protocol on the node which understands that name. For example, an SMS-capable device regards a phone number "name" as an "address", but a non-SMS-capable device would not. This allows for the fact that, as a message moves between nodes, different methods of mapping names to transmission methods become available. Thus a layered model of name-to-address mapping is not always appropriate.

I note that while a given node may need to "resolve" a name further in order to effect sending to that name, e.g. taking an email address, discovering the SMTP server suitable, and using IP routing and a next-hop MAC address to send towards that SMTP server, it is not sensible to regard these looked-up values as members of the name graph, since another node with the same message would need completely different values, and would have to resolve them independently.

While persistent information is stored in name graphs, transient information is captured using the notion of a Neighbor, which identifies the next-hop in the path for the data in order to reach the name. Neighbors are discovered by Connectivities (this is discussed in the next section), and their properties are used by Protocols to establish what names a given communication can be forwarded to.

Name Objects

Haggle represents name graphs using DOs with a particular attribute containing the name as a string, which are known as Name Objects (NOs). These are linked using the normal DO linking function to provide name graphs. The use of DOs for naming allows names to be easily managed and made persistent.

Before Haggle can send data to a name graph, the NOs and links must be constructed somehow. There are many potential sources for NOs. Firstly, although the name graph concept seems at odds with existing user devices, actually much of what is proposed is simply a consolidation of naming information from disparate sources already present on a device. For instance, name-to-email-address mappings, name-to-instant-messaging-ID mappings, etc are already kept by the respective applications. In addition, names can also be gathered from Connectivities such as Bluetooth or 802.11, as MAC addresses of nearby Haggle-capable nodes are regarded as Names. A third discovery method follows on from this, whereby the Name Manager can detect nearby Haggle nodes (via the existence of neighbours) and directly send them a message containing both the NO graph corresponding to the node, and a DO filter requesting the recipient's own NO graph (analogous to the node saying "Hi stranger, I'm Bob, who are you?"). A fourth discovery method is in the receipt of messages where the node is acting as a courier, or is the destination. These messages' NO graphs can be mined for information. A fifth method may be for name graphs to be maintained and distributed during periods of stable connectivity with a trusted server on the Internet.

Finally, it is worth noting that a name graph used as an address for a message need not remain static. Intermediate nodes could potentially add to this graph, either adding hints for forwarding algorithms to perform better routing, e.g. "This MAC address was seen recently", or filling in missing sections of the graph. As an extreme example, a user might be told the name of someone that they wish to send a message to, but not have any other information such as their email address. By simply using the person's name, a message can be created which can only be delivered by (controlled) broadcasting. However, one of the nodes in the room might have the name graph corresponding to that name, and could add the necessary NOs as destinations for the message so that it can be delivered properly.

A.5.3 Connectivity Manager and Connectivities

Haggle aims to support and embrace the use of many different networking technologies at the same time. Networking technologies differ by their range, latency, bandwidth, infrastructure available, cost of using the infrastructure, battery consumption, availability, and so on. It is therefore appropriate for different Connectivities to be used depending on the particular type of communication being sent, e.g. a small but urgent message could use (relatively) expensive GPRS, while large, non-urgent data could wait until a free connection opportunity arises (either locally or via a "free" access point).

The job of the Connectivity Manager in Haggle is simply to encapsulate a number of Connectivity objects and to initialise the appropriate number at start-of-day. Each Connectivity must support a well-defined interface including functionality for neighbor discovery, opening/using/closing communications channels, and estimating the costs (in terms of money, time and energy) of performing network operations. The Connectivity must interface with the underlying hardware to provide this functionality.

There will be one Connectivity instance per instance of a network interface on a node (so if there were two 802.11 interfaces there would be two Connectivities created during initialisation). This is because a Connectivity is regarded by the Resource Manager as a schedulable resource, so it must be clear exactly how many resources there are. Since the Resource Manager expects to schedule the network interface, all operations that result in network activity, including operations initiated by the Connectivity's code itself, must be passed to the Resource Manager as "Tasks" (to be discussed in Section A.5.6).

Connectivities also interact with Protocols, providing them with Neighbor lists gained during neighbor discovery. A Neighbor is a potential next-hop by which particular Protocols may know how to send data of particular types to particular NOs. We differentiate between "non-Internet" Neighbors which are direct next hops running Haggle, and "Internet" Neighbors which are next hops supporting IP for accessing the Internet. Typically, each Protocol will only be interested in one type of Neighbors.

Neighbor discovery can take various forms. In 802.11, any node with reception turned on can see beacons from access points (APs) which announce their existence. For Bluetooth, neighbor discovery is an active (and time-consuming) process. For GPRS, neighbor discovery is implicit in that when base station coverage is present, an Internet Neighbor is visible.

A.5.4 Protocol Manager and Protocols

The Protocol Manager is only responsible for encapsulating a set of Protocols, and initializing that set at start-of-day. A Protocol is a method by which DOs can be forwarded to Names, e.g. SMTP, HTTP, a direct peer-to-peer protocol etc. This highlights an architectural difference between Haggle and traditional network stacks, since these protocols are normally at the application layer and forwarding decisions are normally considered to be taken at the IP layer underneath.

Each Protocol monitors the Neighbors visible through the Connectivities, and using these Neighbors it can determine which NOs it can deliver to. This decision can also take into account the type of data being forwarded, e.g. an SMTP protocol can send a message to an email NO, but it may refuse to accept non-message data (e.g. application signaling) since that is not suitable to appear in an inbox.

For Protocols which must accept incoming connections, e.g. a direct peer-to-peer protocol, they must provide each Connectivity with enough information so that it can redirect incoming data to that protocol. This is akin to listening sockets in the existing architecture. Some Protocols do not accept incoming connections; typically, all Internet-using Protocols (HTTP, SMTP, POP) act as clients to existing servers and so must initiate connections themselves. While seemingly simpler, this proves a source of additional work due to polling requirements — for example, the POP Protocol must use Resource Manager "Tasks" in order to request that email accounts be checked (if Internet connectivity is available).

A.5.5 Forwarding Manager and Algorithms

The Forwarding Manager provides an API to applications to cause data to be sent remotely, encapsulates a number of Forwarding Algorithms, and sends the Forwarding Tasks that are produced by them to the Resource Manager.

Applications request data transfers by specifying a set of DOs (the heads of a larger set of linked DOs) and a set of NOs (the heads of name graphs). The Forwarding Manager constructs a Forwarding Object (FO) which is a DO with metadata about the forwarding operation, and which is linked to the destination NO graphs, to the DOs, and to an NO graph describing the sender (useful for replies either from applications or from Haggle's internal replies with DOs matching DO filters). The metadata can include expiry times and expiry hop counts, security information and routing hints for forwarding algorithms (described below), as well as a list of NOs to which the data has already been sent. Other metadata can also be present — the DO format allows for simple extensibility, and unknown fields can be easily ignored across different implementations/instantiations of Haggle.

Forwarding Algorithms

Once an FO is created, it is the job of one or more Forwarding Algorithms to determine suitable next hops. In Haggle, we precisely define the role of a Forwarding Algorithm as: for each FO, propose a set of {Protocol, NO, Neighbor} tuples which this FO could be sent to, and a scalar "forwarding benefit" associated with each tuple, which is an estimate of the probability that sending it that way would result in successful end-to-end delivery. The tuples and benefit levels change continuously, depending on the available connection opportunities, the known information about the FO (e.g. if it expires or has already been delivered), etc.

Haggle has the useful feature of allowing many Forwarding Algorithms to be in use *simultaneously*. Note that I do *not* mean that traffic is generated according to the wishes of all Forwarding Algorithms, since the Resource Manager will be responsible for accepting or denying the proposed actions of every algorithm. The simplest algorithm is a direct forwarding algorithm, which only proposes to send an FO if it can directly reach an NO which is present in its graph of destinations (i.e. it does not make use of any multi-hop communication), with a forwarding benefit of 100% by definition. Another algorithm is the epidemic forwarding algorithm [VB00], which sends the data to every Name that is reachable, i.e. it floods the data, but with a correspondingly low forwarding benefit. Haggle can also make use of MANET algorithms such as geographic [MWH01] or distance-vector [PBRD03], as well as opportunistic store-and-forward [SRJB] [ZAZ04] such as mobility-based [HCS$^+$05, AAO03, LFC06] algorithms. Haggle is able to use many of these algorithms simultaneously, obtaining the "best of many worlds" in that for each forwarding operation, a different Forwarding Algorithm may prove best, due to availability or not of per-algorithm state information. Such state information can be exchanged in Haggle by Forwarding Algorithms themselves creating Forwarding Tasks targeted at nearby nodes.

For each FO, and each Protocol, Name, Neighbor that an FO is proposed to be sent to (with associated benefit), the Forwarding Manager creates a "Forwarding Task", to be executed when the Resource Manager decides on doing so. When executed, the Forwarding Task causes the associated Protocol to send the FO to the NO, via the Neighbor.

A.5.6 Resource Manager using Tasks

As referred to many times above, all outgoing or incoming network operations in Haggle are proposed to the Resource Manager and executed only if/when the Resource Manager chooses; they are not necessarily executed in order or at all. A Task comprises a method of accomplishing the work, the benefits of achieving the Task, and the costs of performing the Task. This definition is deliberately abstract so that the Resource Manager can compare between different possible actions while knowing little about the actual mechanisms or details of formulating or carrying out Tasks.

Both the costs and benefits of Tasks are re-evaluated by the Resource Manager each time a Task is considered for execution, using callback functions provided at Task creation time. A Task's cost describes measurable, true costs to the node, expressed in terms of energy, money and time-on-network. Time-on-network refers to connectivity-specific nature of the Tasks being scheduled, and represents the opportunity cost of not doing something different with that time. Task benefits describe the estimated utility to end users of executing a Task. This is not a simple calculation to make. Components of this benefit include forwarding benefit which, as described above, is the likelihood that this action will result in a successful end-to-end transmission, but also application benefit (how worth it to the application is that transmission), and user benefit (what priority is the action to the user). We would also like to be able to take into account priorities specified by the owner of the devices, e.g. "I don't want to spend money on others' traffic, but I will allow Haggle to share a limited percentage of my battery")

Tasks can be either asynchronous or immediate. Asynchronous Tasks are the norm, and (as the name suggests) the Resource Manager is provided with a callback to asynchronously call when it wants the Task performed. Asynchronous Tasks can be "persistent", i.e. once they are executed they persist to be executed again later — otherwise, a Task is only called at most once. Immediate Tasks are used when a particular operation must either be done now or not at all, and they are used to deal with events such as incoming network connections which must either be accepted or rejected.

Benefits and Costs for asynchronous Tasks are often varied by their owner over time. For persistent Tasks, the benefit of a Task that has just been executed will be low, e.g. checking for new email on a server or checking for new Neighbours is not that beneficial if the operation was last performed 1 second ago, and the benefit would rise over time. On the other hand, an FO with an imminent application-set expiry time becomes less and less beneficial to forward in a multi-hop fashion, since the likelihood of reaching the destination in time for the application purpose becomes low. Costs are calculated with the assistance of the Connectivity that the Task will be using — typically, the Task owner would provide an estimate of the number of bytes to be sent/received via a particular Neighbor, and the Connectivity can translate that into the expected money, energy, and time that this transmission will take.

Once a Task is being executed, the Resource Manager can also be asked for a “continuation”, i.e. if the scope of the work being done by the Task increases over the initial cost/benefits specified, then the Resource Manager can be synchronously polled for permission (with additive cost/benefit over the existing Task) to authorise work on the extended Task. This is useful for circumstances such as email checking, which may discover a large email waiting for download.

The Task model is in marked contrast to the traditional network stack, where networking operations proposed by applications or operating system functions are always attempted. The centralisation of decision-making about what Tasks are worth doing at all, and which are more important at any time, allows Haggle to have a number of advantageous features.

The Resource Manager is a key illustration of how Haggle is “layerless” since Tasks come from many different managers. I have already described examples of Tasks generated by the Name Manager (querying nearby Neighbors for name information), Forwarding Manager (exchanging state information), Protocols (email checking), and Connectivities (Neighbor discovery).

In the current version of Haggle, security and privacy have not been addressed as key concerns. I have briefly discussed them in Section 1.7, but no real implementation has be done. I will strongly raise these important issues in the next version of the prototype.

A.6 Comparison with the DTN Architecture

DTN [Fal03] is a network architecture designed for challenged networks, which are networks that may violate one or more of the three key assumptions about the underlying link characteristics: an end-to-end path exists between a data source and its peer(s), the maximum round-trip time between any node pairs in the network is not excessive, and the end-to-end packet drop probability is small. Examples of these networks includes terrestrial mobile networks, exotic media networks, military Ad-Hoc networks, and sensor networks. Clearly, PSN is one kind of challenged network and hence it falls under DTN.

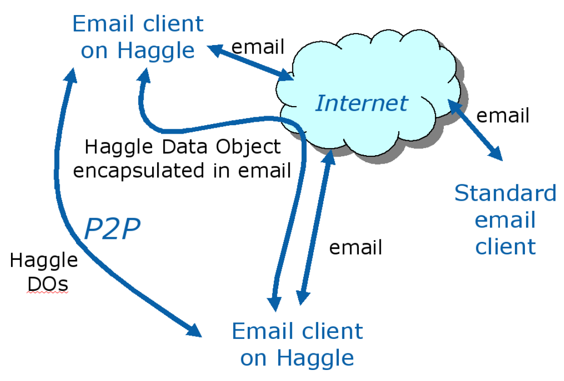
As a proposed architecture for PSN, Haggle shares some design principles with DTN, such as both of them use application-level message switching, late name binding, and custody-persistent storage for store-and-forward. For message switching, DTN uses Application-level Data Units (ADUs) and Haggle uses DOs. In DTN, ADUs are typically sent by and delivered to applications in complete units. They are transformed by the bundle layer into one or more protocol data units called “bundle” for forwarding, bundles may be further fragmented during transmission. The data-centric design of Haggle (DO) enables them to be used by many parties. This design approach is driven by “ad hoc Google” applications.

Because of the data-centric principle and application characteristics of PSNs, Haggle has clear architectural differences from the DTN architecture. DTN adopts an overlay architecture, and Haggle uses the layerless concept. DTN aimed to provide interoperability for different networks such as the Internet, Bus networks, and exotic media networks, it has to be designed as an overlay architecture. It laid on top of the transport layer or other network layer. It relies on the implementation of the convergence layer to adapt the bundle layer to the underlying transport layer. Different underlying protocols i.e. TCP and UDP, need a corresponding convergence layer implementation. Because DTN only has specifications for its bundle layers but no specification for the underlying network type, the underlying network type can be sockets, sensor networks and Haggle.

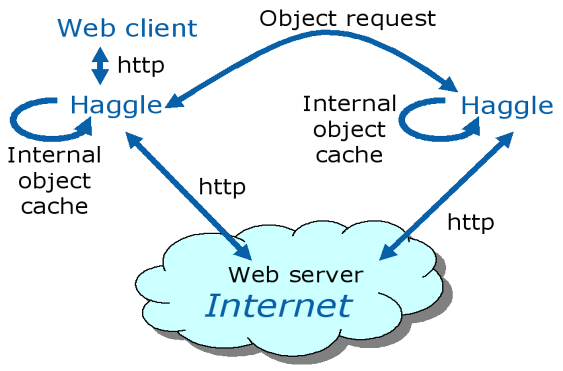
Haggle is a clean-slate design tailor-made for mobile devices based on the characteristics of mobile networks (i.e. lack of end-to-end connectivity, availability of local peer-to-peer), and expected applications such as asynchronous messaging, web-browsing, and ad hoc Google (one reason for taking a data-centric approach). Considering the Internet as a useful medium for relaying data for mobile device communication, Haggle incorporates popular Internet application protocols such as SMTP, POP, and HTTP into its Protocol manager, so Haggle can talk to the Internet directly, instead of using proxies. But unlike DTN, Haggle did not aim to be a universal interoperability platform to link different networks from sensors to near-Earth satellite communications. It is specific to mobile devices. DTN can run on top of Haggle by implementing a Haggle convergence layer, or Haggle can provide the DTN protocol in the protocol manager for interoperability. Because these two architectures share the same vision about challenged network characteristics and some key design principles, it is easy for them to be compatible with each other, and actually Haggle can be a clean-slate layerless implementation reference for DTN.

A.7 Short Summary of Prototyping

We have developed a Haggle prototype using Java initially, targeting Windows XP. This development has been conducted using *sourceforge.net*, an open source development site, under the GNU General Public License (GPL), and remains open-source and available to other research groups. The implementation of the prototype is not a main contribution of this dissertation, so



(a) Email Application



(b) Web Application

Figure A.4: Haggle Email and Web Applications

I will not go into the details here, but give a brief summary for the completeness of this work.

We implemented Haggle's Data Manager using Java's standard SQL interface, backed by MySQL. Although any SQL back-end will work, we chose MySQL because it is freely available. We chose SQL as the storage mechanism for the Data Manager because it provides an easy interface for persistence and search. DO Filters are implemented as regular expression-like queries, and for the most part these queries can be translated directly into SQL select statements, to take full advantage of SQL's optimised environment. We chose to focus on 802.11 connectivity for our prototype. This is because it is a widely used wireless access network, and is available for a range of devices from laptops to mobile phones. It also offers both neighborhood and infrastructure connections (through ad hoc mode and infrastructure mode respectively) which allow us to explore the range of Haggle capabilities using a single connectivity type. Implementing 802.11 support requires a native driver component which communicates with the NDIS driver interface for Windows XP. We implemented this in C++ resulting in a dll file, which provides our Java code with capabilities such as putting the 802.11 interface in ad hoc mode or AP mode. We implemented two forwarding algorithms so far, namely direct and epidemic. The direct algorithm only proposes to send an FO to a NO if that NO appears as a destination of the FO. The epidemic algorithm proposes to send every FO to every NO where a Protocol says it can support that transfer, but does so with a lower forwarding benefit since it has no idea whether it will reach the destination this way.

Based on our introductory motivating examples, we have chosen to target email and the web as our prototype applications. To be clear, by “email” and “web” we mean the messaging and hyperlinked-information applications, rather than the protocols that underly them. Both of these applications enjoy huge support from the pre-existing infrastructure deployment of servers and content. It is a crucial feature of Haggle that we can take advantage of this infrastructure as well as providing new functionality (operation when infrastructure is not available). This makes Haggle much more compelling to existing users of that infrastructure, and the value added by Haggle provides motivation for its deployment. To provide legacy support for existing email and web applications, we implement localhost SMTP/POP and HTTP proxies alongside Haggle. This allows users to keep using the same applications they habitually use (we have tested Outlook Express, Thunderbird, Internet Explorer and Firefox) with only minimal reconfiguration. The modes of communication and operation of email and web applications using Haggle are highlighted in Figure A.4. For the details of the prototype, the readers may refer to the separate technical report [SSH$^+$07].

A.8 Conclusion

Haggle is a clean-slate node architecture for mobile devices. Haggle allows applications to become infrastructure-independent, freeing them from having to explicitly handle different and changing connectivity environments. I prototype Haggle using existing email and web applications, showcasing their ability to operate in ad hoc networking circumstances where they would previously have failed. This allows people to use the same application across different connectivity scenarios, something they cannot do today without manual configuration.

Haggle provides a uniform interface for exposing application-layer names and naming metadata to allow late-binding of data delivery. This allows Haggle to select the best protocols and connectivities to use, under any given network constraints, for reaching the destination. The Resource Manager provides a single informed decision point for managing the usage of network resources, allowing the node to coordinate the needs of its applications with the user's preferences.

On the another hand, this architecture and prototype work points us to an important issue of PSN, the forwarding algorithms, which led to the main contributions of this thesis, social-based forwarding. At this moment I only implement direct transfer (wait for destination) and epidemic forwarding for the prototype, I want to implement and test my BUBBLE algorithm on the Haggle testbeds in the near future.

Bibliography

[AAO03] A.Lindgren, A.Doria, and O.Schelen. Probabilistic routing in intermittently connected networks. *SIGMOBILE MCCR*, 7(3):19–20, 2003. (pg 148)

[AB02] Reka Albert and Albert-Laszlo Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002. (pg 92)

[AHLP01] Lada A. Adamic, Bernardo A. Huberman, Rajan M. Lukose, and Amit R. Puniyani. Search in power law networks. *Physical Review E*, 64:46135–46143, October 2001. (pg 105)

[ASS$^+$] Marcos K. Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar D. Chandra. Matching events in a content-based subscription system. In *Proceedings of PODC '99*. (pp 16, 138)

[AWSBL] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *Proceedings of SOSP 1999*. (pp 17, 139, & 144)

[BB03] F. Baccelli and P. Bremaud. *Elements of Queuing Theory*. Springer-Verlag, second edition, 2003. (pg 29)

[Bol01] Béla Bollobás. Random graphs. 2001. (pp 92, 97)

[Bre99] P. Bremaud. *Markov Chains, Gibbs Field, Monte Carlo Simulation and Queues*. Springer-Verlag, first edition, 1999. (pg 29)

[CDT06] Edward Cutrell, Susan T. Dumais, and Jaime Teevan. Searching to eliminate personal information management. *Commun. ACM*, 49(1):58–64, 2006. (pg 141)

[CHC$^+$06] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proc. INFOCOM*, April 2006. (pp 20, 82, 94, 98, & 99)

[CHC$^+$07] Augustin Chaintreau, Pan Hui, Jon Crowcroft, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007. (pp 22, 36)

[CJMW05] Hyunseok Chang, Sugih Jamin, Z. Morley Mao, and Walter Willinger. An empirical approach to modeling inter-as traffic matrices. In *Proceedings of ACM Internet Measurement Conference*. ACM, 2005. (pg 97)

[CJW06] Hyunseok Chang, Sugih Jamin, and Walter Willinger. To peer or not to peer: Modeling the evolution of the internet's as-level topology. In *Proceedings of IEEE Infocom*. IEEE, 2006. (pp 92, 97)

[Cla88] D. Clark. The design philosophy of the darpa internet protocols. In *ACM SIGCOMM CCR*, pages 106–114, 1988. (pg 135)

[Cla05] Aaron Clauset. Finding local community structure in networks. *Physical Review E*, 72:026132, 2005. (pp 14, 67, 68, & 71)

[CLP06] Paolo Crucitti, Vito Latora, and Sergio Porta. Centrality measures in spatial networks of urban streets. *Physical Review E*, 73:036125, 2006. (pg 130)

[CM01] X. Chen and A.L. Murphy. Enabling disconnected transitive communication in mobile ad hoc networks. In *Proceedings of the Workshop on Principles of Mobile Computing*, pages 21–27, 2001. (pg 37)

[CNM04] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70:066111, 2004. (pp 50, 55, 56, & 57)

[D$^+$04] C. Diot et al. Haggle project,http://www.haggleproject.org, 2004. (pg 67)

[DDDGA05] Leon Danon, Jordi Duch, Albert Diaz-Guilera, and Alex Arenas. Comparing community structure identification, 2005. (pp 14, 49, 73, & 97)

[DFGV03] Henri Dubois-Ferriere, Matthias Grossglauser, and Martin Vetterli. Age matters: Efficient route discovery in mobile ad hoc networks using encounter ages. In *Proc. MobiHoc'03*, 2003. (pg 45)

[DFL01] James A. Davis, Andrew H. Fagg, and Brian N. Levine. Wearable computers as packet transport mechanisms in highly-partitioned ad-hoc networks. In *ISWC '01: Proceedings of the 5th IEEE International Symposium on Wearable Computers*, page 141, 2001. (pg 37)

[DH07] Elizabeth Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of ACM MobiHoc*, 2007. (pp 18, 117, & 125)

[Dun98] Robin Dunbar. *Grooming, Gossip, and the Evolution of Language*. Harvard Univ Press, October 1998. (pg 84)

[EP06] Nathan Eagle and Alex Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, V10(4):255–268, May 2006. (pp 20, 27, 67, & 99)

[Fal03] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. SIGCOMM*, 2003. (pp 12, 17, 134, & 149)

[FLGC02] Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans Coetzee. Self-organization of the web and identification of communities. *IEEE Computer*, 35(3):66–71, 2002. (pg 48)

[Fre77] L. C Freeman. A set of measuring centrality based on betweenness. *Sociometry*, 40:35–41, 1977. (pp 14, 49, 98, & 100)

[GT02] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. *IEEE/ACM Trans. on Networking*, 10:477–486, 2002. (pp 18, 38, 43, 44, & 125)

[HC07] P. Hui and J. Crowcroft. How small labels create big improvements. In *Proc. IEEE ICMAN*, March 2007. (pp 98, 107, & 125)

[HCG$^+$05] P. Hui, A. Chaintreau, R. Gass, J. Scott, J. Crowcroft, and C. Diot. Pocket switched networking: Challenges, feasibility, and implementation issues. In *Proc. WAC*, 2005. (pg 134)

[HCS$^+$05] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proc. WDTN*, 2005. (pp 20, 22, 23, 37, 82, 83, 94, 99, & 148)

[HHLM99] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402(6761 Suppl), December 1999. (pg 48)

[HKA04] T. Henderson, D. Kotz, and I. Abyzov. The changing usage of a mature campus-wide wireless network. In *Proc. Mobicom*, 2004. (pp 20, 22, 27, & 80)

[HYyCC07] Pan Hui, Eiko Yoneki, Shu yan Chan, and Jon Crowcroft. Distributed community detection in delay tolerant networks. In *To Appear in Sigcomm Workshop MobiArch '07*, August 2007. (pg 118)

[Jac01] P. Jaccard. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547, 1901. (pp 68, 74, & 120)

[JLW05] E. P. C. Jones, L. Li, and P. A. S. Ward. Practical routing in delay-tolerant networks. In *Proc. WDTN*, 2005. (pg 12)

[J.N04] M. E. J.Newman. Analysis of weighted networks. *Physical Review E*, 70:056131, 2004. (pp 47, 48, 50, 56, 57, 59, & 77)

[JOW$^+$02a] H Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early expe riences with zebranet. In *Proc. ASPLOS-X*, 2002. (pg 22)

[JOW$^+$02b] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 96–107, 2002. (pg 43)

[JSB$^+$04] S. Jain, R. C Shah, G. Borriello, W. Brunette, and S. Roy. Exploiting mobility for energy efficient data collection in sensor networks. In *Proceedings of IEEE WiOpt*, 2004. (pg 43)

[KHA04] David Kotz, Tristan Henderson, and Ilya Abyzov. CRAWDAD data set dartmouth/campus (v. 2004-12-18). Downloaded from http://crawdad.cs.dartmouth.edu/dartmouth/campus, December 2004. (pg 23)

[KKK02] David Kempe, Jon Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. *J. Comput. Syst. Sci.*, 64(4):820–842, 2002. (pg 14)

[KKP] Martin Karsten, S. Keshav, and Sanjiva Prasad. An axiomatic basis for communication. In *Proceedings of HotNets 2006*. (pg 144)

[Kle06] Jon Kleinberg. Complex networks and decentralized search algorithms. 2006. (pg 39)

[LDS04] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *Proc. SAPIR*, 2004. (pp 12, 18, 21, 45, 82, 98, 116, & 125)

[Leb05] *Knowledge-Based Opportunistic Forwarding in Vehicular Wireless Ad Hoc Networks*, volume 4, 2005. (pp 18, 117, & 125)

[LFC06] J. Leguay, T. Friedman, and V. Conan. Evaluating mobility pattern space routing for DTNs. In *Proc. INFOCOM*, 2006. (pp 18, 45, 117, 125, & 148)

[LLS$^+$06] Jeremie Leguay, Anders Lindgren, James Scott, Timur Friedman, and Jon Crowcrof. Opportunistic content distribution in an urban setting. In *ACM CHANTS*, pages 205–212, 2006. (pg 99)

[LN04] David Lusseau and M. E. J. Newman. Identifying the role that individual animals play in their social network. *PROC.R.SOC.LONDON B*, 271:S477, 2004. (pg 48)

[Mar02] Peter V. Marsden. Egocentric and sociocentric measures of network centrality. *Social Networks*, 24(4):407–422, October 2002. (pp 126, 130)

[MGD06] Alan Mislove, Krishna P. Gummadi, and Peter Druschel. Exploiting social networks for internet search. In *Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets'06)*, November 2006. (pg 15)

[MHM05] M. Musolesi, S. Hailes, and C. Mascolo. Adaptive routing for intermittently connected mobile ad hoc networks. In *Proc. WOWMOM*, 2005. (pp 18, 117, & 125)

[Mil77] Stanley Milgram. *The individual in a social world: essays and experiments*. Addison-Wesley Pub. Co., Reading, Mass., 1977. (pg 14)

[MM06] Mirco Musolesi and Cecilia Mascolo. A Community based Mobility Model for Ad Hoc Network Research. In *Proceedings of the 2nd ACM/SIGMOBILE International Workshop on Multi-hop Ad Hoc Networks: from theory to reality (REALMAN'06)*. ACM Press, May 2006. (pg 82)

[MMG$^+$07] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 5th ACM/USENIX Internet Measurement Conference (IMC'07)*, October 2007. (pp 95, 130)

[MV05] Marvin Mcnett and Geoffrey M. Voelker. Access and mobility of wireless pda users. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(2):40–55, April 2005. (pp 20, 27)

[MWH01] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network*, 15(6), Nov 2001. (pg 148)

[New04a] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69:066133, 2004. (pg 73)

[New04b] M.E.J. Newman. Detecting community structure in networks. *Eur. Phys. J. B*, 38:321–330, 2004. (pp 14, 48, 49, & 97)

[New05] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46:323, 2005. (pg 23)

[New06] M. E. J. Newman. Modularity and community structure in networks. *PROC.NATL.ACAD.SCI.USA*, 103:8577, 2006. (pp 14, 97)

[NG04] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, February 2004. (pp 14, 49, 50, 55, 57, & 97)

[NSDG06] M. Piorkowski N. Sarafijanovic-Djukic and M. Grossglauser. Island hopping: Efficient mobility-assisted forwarding in partitioned networks. In *IEEE SECON*, 2006. (pp 18, 125)

[Oka05] Samir Okasha. Altruism, group selection and correlated interaction. *British Journal for the Philosophy of Science*, 56(4):703–725, December 2005. (pp 13, 81, 83, & 98)

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. *RFC3561*, 2003. (pg 148)

[PDFV05] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005. (pp 47, 48, 50, 60, 71, & 97)

[PG04] E. Paulos and E. Goodman. The familiar stranger: Anxiety, comfort, and play in public places. In *Proc. ACM SIGCHI*, 2004. (pg 53)

[RB06a] Joerg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74:016110, 2006. (pg 97)

[RB06b] Martin Rosvall and Carl T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks, Dec 2006. (pg 97)

[RCC$^+$04] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *PROC NATL ACAD SCI USA*, 101:2658, Feb 2004. (pg 49)

[RSM$^+$02] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabasi. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, August 2002. (pg 48)

[Sal89] Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. (pg 120)

[SAZ$^+$] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. In *Proceedings of SIGCOMM 2002*. (pg 143)

[SCP$^+$04] Jing Su, Alvin Chin, Anna Popivanova, Ashvin Goel, and Eyal de Lara. User mobility for opportunistic ad-hoc networking. In *WMCSA '04: Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*, pages 41–50, Washington, DC, USA, 2004. IEEE Computer Society. (pg 27)

[SH03] Tara Small and Zygmunt J. Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 233–244, 2003. (pg 22)

[SHCD06] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Haggle: a networking architecture designed around mobile users. In *Proc. WONS*, 2006. (pp 43, 133, & 138)

[SMS06] G. Sharma, R. Mazumdar, and N. Shroff. Delay and capacity trade-offs in mobile ad hoc networks: A global perspective. In *Proceedings of IEEE Infocom*, 2006. (pp 43, 44)

[SOR04] Nima Sarshar, P. Oscar, and Vwani P. Roychowdhury. Percolation search in power law networks: making unstructured peer-to-peer networks scalable, 2004. (pg 124)

[SPR05] T. Spyropoulos, K. Psounis, and C. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile n etworks. In *Proc. WDTN*, 2005. (pp 18, 125)

[SRJB] Rahul C. Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Datamules: Modelling a three tiered architecture for sparse sensor networks. In *IEEE SNPA 2003*. (pg 148)

[SSH$^+$07] Jing Su, James Scott, Pan Hui, Eben Upton, Meng How Lim, Christophe Diot, Jon Crowcroft, Ashvin Goel, and Eyal de Lara. Haggle: Clean-slate networking for mobile devices. Technical Report UCAM-CL-TR-680, University of Cambridge, Computer Laboratory, January 2007. (pg 152)

[TMMS04] Ala Trusina, Sergei Maslov, Petter Minnhagen, and Kim Sneppen. Hierarchy measures in complex networks. *Physical Review Letters*, 92:178702, 2004. (pg 106)

[TTP$^+$95] Douglas B. Terry, Marvin M. Theimer, Karin Petersen, Alan J. Demers, Mike J. Spreitzer, and Carl H. Hauser. Managing update conflicts in Bayou, a weakly connected replicated storage system. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP-15), Copper Mountain Resort, Colorado*, 1995. (pg 16)

[VB00] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, April 2000. (pp 18, 37, 43, 125, & 147)

[Wat99] Duncan J. Watts. *Small Worlds – The Dynamics of Networks between Order and Randomneess*. Princeton University Press, Princeton, New Jersey, 1999. (pg 54)

[WFI94] Stanley Wasserman, Katherine Faust, and Dawn Iacobucci. *Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, November 1994. (pg 49)

[Win60] P.R. Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6:324–342, 1960. (pg 119)

[YCM06] Alexander Yip, Benjie Chen, and Robert Morris. Pastwatch: a distributed version control system. In *NSDI'06: Proceedings of the 3rd conference on 3rd Symposium on Networked Systems Design & Implementation*, pages 28–28, Berkeley, CA, USA, 2006. USENIX Association. (pg 16)

[YHCC07] Eiko Yoneki, Pan Hui, ShuYan Chan, and Jon Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *MSWiM '07: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, pages 225–234, New York, NY, USA, 2007. ACM. (pg 129)

[ZAZ04] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 187–198, New York, NY, USA, 2004. ACM Press. (pp 18, 125, & 148)