Collaborative Routing in Mobile Partitioned Networks

THÈSE N^{O} 4599 (2010)

PRÉSENTÉE LE 23 FÉVRIER 2010

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 4
SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Natasa SARAFIJANOVIC-DJUKIC

acceptée sur proposition du jury:

Prof. S. Süsstrunk, présidente du jury
Prof. M. Grossglauser , directeur de thèse
Prof. E. M. Belding, rapporteur
Prof. J.-Y. Le Boudec, rapporteur
Dr C. Mascolo, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2010

In memory of my dear sister Vasa

Acknowledgments

First of all, I would like to thank my advisor, Professor Matthias Grossglauser. I am grateful to him for all his time he spent guiding and contributing to this work and for interesting teaching-assistant tasks I did for his courses. In addition, I want to thank other LCA professors, Jean-Yves Le Boudec, Patrick Thiran, and Jean-Pierre Hubaux, for their advice and courses. I would also like to thank the members of my thesis committee, Cecilia Mascolo, Elizabeth Belding, and Jean-Yves Le Boudec, for their time, availability and helpful feedback, and Sabine Süsstrunk for presiding over the defense.

Next, I would like to thank colleagues from the LCA lab as well as the Doctoral School group, for all their help and useful information. I am especially grateful to my LCA4 colleagues. Special thanks go to Michal Piorkowski for a fruitful collaboration in a joint project, whose contribution justified an adjective realistic to my work. Also, I thank Henri Dubois-Ferrière for help with the first steps of the learning the NAB network simulator, Dominique Tschopp for translating my thesis abstract, and Daniel R. Figueiredo and Pedram Pedarsani for their feedback on my presentations. I also thank the LCA4 members for interesting LCA4 ski outings. In addition, I thank to my office mates Imad Aad, Etienne Perron, and Dominique Tschopp, for giving me a feeling that I am not all alone in solving often unsolvable problems.

I am grateful to the LCA system managers Marc-André Lüthi, Jean-Pierre Dupertuis, Yves Lopes, and Hervé Chabanel, and the LCA secretaries Danielle Alvarez, Holly Cogliati, Patricia Hjelt and Angela Devenoge, for making things a lot easier. Special thanks go to Holly for making sure that my english writings become nice.

Further, I would like to thank everyone that helped me with taking care of my child Marko, which allowed me to concentrate on my work. Special thanks go to the POLYCHINELLE and POLIKIDS group at EPFL for giving such a nice and friendly environment to my child, and for their willingness to accommodate to some of our requirements. I also thank Marko's private babysitters Milena, Jasna, Mira, Milka, Jelena and Bozica, for helping with both household duties and taking care of Marko - And I thank to my family and my husband's family for playing with Marko during the vacation breaks. In addition, I thank Nenad and Vesna, Vlada and Jelena, Gisela and Hicham, and Gnana and Vidhya, the parents of Marko's friends, for sharing their experiences and useful information, and for having nice outings

with our children.

I would also like to take this opportunity to express my gratitude to my “health advisors”, Dr. Douglas Graham (http://foodnsport.com), Esther Joy van der Werf (http://www.visionsofjoy.org), Milija Petrovic and Claude Newell-Lesslauer. Their advices were invaluable in the last years of my thesis when I got into serious health problems.

Finally, my gratitude goes to my family. I thank my parents, Dragica and Jovan, for their emotional and financial support throughout my education, without which my undergraduate studies would be impossible. I am also very grateful to my sisters, Vasa and Veronika, for beautiful memories of an always interesting childhood. These childhood memories helped me a lot not to give up my thesis work when the serious difficulties arose. Unfortunately, during my PhD work something very sad happened, my younger sister Vasa lost her life. She was a very special person to me, and I dedicate this thesis to her. At last, but not least, I thank my husband Slavisa for all his love and support during our PhD years - And I thank Marko, our child, for making me smile when everything else seemed unsmiling.

Abstract

Embedded wireless networks find a broad spectrum of applications in transportation, environmental monitoring, logistics, supply chain management, and ”pocket-switched” communication. The node mobility patterns in these applications tend to give rise to spatially heterogeneous node distributions, which may cause network partitions. In this thesis, we consider the problem of routing in mobile networks under such challenging conditions. More specifically, we endeavor to identify features of mobility common to different applications, in order to devise routing methods that are tailored to exploit these features. We explore two features in particular, (i) predictability and (ii) stable and heterogeneous spatial node distribution.

A mobility process is predictable if the future location of a node can be well estimated, given knowledge of its current and past locations and possibly other statistics. We show how the performance of routing can be improved by explicitly incorporating mobility prediction. Specifically, we consider the performance of Last Encounter Routing (LER) under a simple synthetic random waypoint (RWP) mobility model. We extend the LER algorithm so that it takes into account predicted node trajectories when making routing decisions, and we show that this significantly improves its performance.

A mobility process has a stable spatial node distribution if, informally, the node density remains the same over time, even though individual nodes are not constrained in space. This is a common feature of many mobility patterns because the spatial distribution is determined by the natural or constructed environment, regardless of the behavior of individual nodes. This typically leads to heterogeneous connectivity and to network partition, where highly connected clusters are interspersed with low-connectivity regions. We model such a situation with a set of stable concentration points (CPs) characterized by high node density, and with a mobility process that describes how nodes move between these islands of connectivity. We study two instances of this model: the G-model, where the CPs and the flow of nodes are abstracted as a graph, and the H-model, where nodes perform heterogeneous random walks on the plane.

We exploit the presence of this stable CP topology in order to develop an efficient routing algorithm under these two mobility models. Our routing algorithm, Island Hopping (IH), exploits knowledge of the CP topology to make routing decisions. IH achieves a very good delay-throughput trade-off compared with several other existing routing algorithms, and it scales well with the network size.

In many situations, it would be unrealistic to assume that CPs and the flows of mobile nodes among them are known a-priori. We develop methods, collectively called Collaborative Graph Discovery (COGRAD), that allow the nodes to discover the CP graph without any explicit signals from the environment (such as GPS coordinates or fixed beacons). We show that COGRAD can replace an oracle with knowledge of the CP topology after a sufficient warm-up period, allowing IH to operate even in scenarios without any cues from the environment.

Keywords

Mobile ad-hoc networks, partitioned networks, routing protocols, mobility modeling

Résumé

Il existe une large palette d'applications pour les réseaux sans fil embarqués, notamment dans la surveillance environnementale, la logistique, la gestion des chaînes d'approvisionnement et les communications pocket switched.

La mobilité des noeuds dans ce type de scénario engendre des distributions hétérogènes, et peut causer des partitions dans le réseau. Dans cette thèse, nous nous intéressons au routage dans ces conditions difficiles. Plus précisément, afin de développer des méthodes de routage adaptées à ces spécificités, nous tentons d'identifier des spécificités propres à toutes les différentes applications ci-dessus. Nous étudions deux propriétés plus particulièrement : (i) la prédictibilité et (ii) la distribution spatiale stable et hétérogène des noeuds.

Un processus de mobilité est prédictible s'il est possible d'estimer une position future en se basant sur la position actuelle, les positions passées et potentiellement d'autres statistiques. Nous démontrons que la performance du routage peut être améliorée en incorporant la prédiction de mobilité. Spécifiquement, nous nous intéressons à la performance de Last Encounter Routing (LER) sous un modèle de mobilité random waypoint (RWP) synthétique simple. Nous modifions LER pour prendre en compte la prédiction des trajectoires des noeuds et démontrons que cela améliore considérablement la performance.

De manière informelle, l'on peut décrire un processus de mobilité spatialement stable comme un processus dont la densité de noeuds reste stable avec le temps, malgré qu'individuellement les noeuds se déplacent et n'ont pas de contraintes spatiales. C'est là une spécificité commune à de nombreux types de mobilité, étant donné que la distribution des noeuds est déterminée par l'environnement et les constructions, et non pas par les déplacements individuels. Cela amène typiquement des distributions hétérogènes et des partitions, où des régions très concentrées et connectées succèdent à des régions très peu peuplées, avec peu de connectivité. Nous modélisons cette situation avec un ensemble de points de concentration (CP) caractérisés par une grande densité de noeuds et un processus de mobilité décrivant les déplacements entre ces îlots de connectivité. Nous étudions deux instances de ce modèle : le modèle G, où les CP et les flux de noeuds sont représentés de manière abstraite dans un graphe, et le modèle H, où les noeuds effectuent des marches aléatoires en deux dimensions.

Nous exploitons la présence de ces CP stables afin de développer un algorithme de routage efficace pour les deux modèles de mobilité. Notre algorithme de

routage, Island Hopping (IH), exploite la connaissance de la topologie des CP afin de prendre des décisions de routage. IH offre un excellent compromis entre le délai et la bande passante en comparaison d'autres algorithmes de routage, et s'adapte bien à de grands réseaux. Dans beaucoup de cas, il ne serait pas réaliste de faire l'hypothèse que les CP et les flux de noeuds sont connus à l'avance.

Nous développons des méthodes, regroupées sous l'appellation Collaborative Graph Discovery (COGRAD) qui permettent aux noeuds de découvrir le graphe de CP, sans signaux extérieurs (tels des coordonnées GPS ou des émetteurs fixes). Nous montrons que COGRAD peut remplacer un oracle connaissant les CP après une période d'échauffement suffisante, permettant ainsi à l'algorithme IH de fonctionner indépendamment de tout signal extérieur.

Mots clés

Réseaux mobiles ad-hoc sans fil, réseaux partitionnés, protocoles de routage, modélisation de la mobilité

Contents

[1]The results presented in this section as well as in Sections 3.2.1 and 3.3.2 are contributions of Michal Piorkowski [Pio09]. These results were obtained during our joint work for the purposes of i) observing important properties of realistic mobility, and ii) validating our modeling of the observed mobility properties. More precisely, finding realistic mobility data sets, the algorithms for the performed data analysis as well as the data analysis are all contributions of Michal Piorkowski. However, mobility modeling is my contribution. We include the results contributed by Piorkowski in order to help a reader in understanding better the presented material.

Chapter 1

Introduction

In the 20th century, mankind have witnessed a development of world-wide communication networks that connect both people and machines. Today we have radio, television, fixed wired telephony, mobile wireless telephony, Internet, wireless LAN. They have become an important part of our everyday lives. Further developments of the technology (such as increasing processing power, extended battery life, and miniaturization) have opened great possibilities for a vast array of new applications. For example, these technological advances make it feasible and affordable to embed wireless communication devices into various other objects. The added communication and networking capabilities to these objects extend their purpose and their usability. So, the embedded wireless networks have a broad spectrum of applications such as transportation, environmental monitoring, wild-life tracking, logistics, supply-chain management, and "pocket-switched" communications.

In these applications, contrary to the conventional global communication networks that are infrastructure-based, an ad-hoc infrastructureless communication paradigm is more suitable. In the conventional networks, infrastructure (such as base stations, access points, switches, routers, etc.) allows devices to communicate over large distances. Whereas, an ad-hoc network is a self-organized network of wireless nodes that may be mobile, where every node can be a router that forwards traffic for other nodes, without (necessarily) using a pre-existing infrastructure. In the above mentioned applications it is often a local communication between nearby devices that matters; thus making an ad-hoc paradigm feasible in such applications. Moreover, a lack of infrastructure allows for relatively low-cost and quick deployment, as well as more flexibility in communication capabilities.

Although ad-hoc networks have received much attention from the research community recently, the idea of ad-hoc networking is not new. In the 1970s the U.S. Defense Advanced Research Projects Agency (DARPA), sponsored the PRNET (Packet Radio Network) project in 1972 [JT87]. This was followed by the SURAN (Survivable Adaptive Radio Network) project in the 1980s [SW87]. These projects dealt with automatic call set-up and maintenance in packet radio networks

with moderate mobility. However, interest in this area grew rapidly in the 1990s due to the advances in wireless communication technology. Since then, ad-hoc networks have been extensively studied.

Real-world implementations are mostly in the research phase. Hence, there are just a few examples in industrial and commercial use. These are mainly in the settings where nodes are static, such as wireless sensor networks and wireless mesh networks. Recently, there have also appeared some practical uses in mobile settings. One Laptop per Child program [lap00] developed an inexpensive laptop computer designed to be distributed to children in developing countries as an educational device. This laptop makes use of an IEEE 802.11s based ad-hoc wireless mesh networking chip. In September 2007, the Swedish company TerraNet AB [ter00] presented a mesh network of mobile phones that allowed calls and data to be routed between participating handsets, without cell sites. Vehicular ad-hoc networks (VANETs) [TML08] are one of the most promising applications of mobile ad-hoc networks, and though not yet in commercial use they are in a very advanced test-field phase.

One of the key network services that needs to be available to applications is routing. This service needs to be able to deliver data from a source node to a destination node, where nodes are identified by their addresses. The design of the routing service in ad-hoc networks faces challenges not present in the infrastructure-based networks. The main challenge is coping with mobility of the nodes. In this thesis, we consider the problem of routing in ad-hoc networks with realistic mobility patterns in possible applications.

The design of routing protocols usually relies on a set of assumptions on the network topology and dynamics. For example, a particular mobility model (random waypoint, random walk, etc.) or a class of network topologies is postulated. If such assumptions are too strong, there is a danger of tailoring the algorithm to these assumptions, instead of designing a robust algorithm which is effective over a wide range of settings. Therefore, we need to carefully check these assumptions, and test candidates against different models to assess their robustness. To do this, we first review the state of the art of both mobility models and realistic mobility traces. Then, we review existing routing mechanisms for mobile ad-hoc networks. Finally, we define our assumptions and modeling of the network topology and dynamics, and we desribe our routing methods designed for these assumptions; thus summarizing contributions of the thesis. We also discuss how our work relates with the state of the art.

1.1 Models of Mobility and Connectivity

Next, we review the existing models of mobility and network connectivity.

Researchers in ad-hoc networking first used random mobility models, such as random walk (RW) [CBD02] and random waypoint (RWP) [JM96a]. In these models, nodes move in a simulation area (e.g., a plane or a disk or a torus) identically

and independently of each other in a random fashion. A node moves all over the simulation area, and its movement is independent both of the node's position and time. Thus, these models lead to homogenous (or close to homogenous) spatial distribution in space.

In order to have a more realistic mobility model, the researchers introduced some diversities in the nodes' movement. We distinguish diversities through space, time and nodes.

The space diversity means that a node's movement depends on its own location. So, there are models where nodes move only in constrained areas, such as roads. They are a city section model [Dav00], Manhattan model, a freeway model[BSH03], and a graph-based model [THB+02]. In an area graph-based model [BRS05], besides the constrained areas, nodes also have different speeds and staying times in different areas. There are also models where a node has preferred locations that it visits more frequently than others. Examples are a weighted waypoint model [HMS+05], a community model [LDS03a], and a clustered model [LYD06]. The work of [JBRAS03] introduces obstacles for the nodes movements.

The time diversity means that a node changes either its mobility model or the parameters of the model periodically through time. An example is [HSPH07]. Note that this model captures all three kinds of the aforementioned diversities.

The node diversity means that nodes move differently among each other (different model or parameters), or that a node's movement depends on other nodes. In this class of models, we have group models (e.g., [HGPC99, SM01]), and social-based models (e.g., [MM07]).

All these models, with the appropriate set-up of their parameters, lead to a heterogenous spatial distribution of nodes. This type of distribution can give rise to heterogeneous connectivity between nodes, where highly connected clusters are interspersed with low-connectivity regions. This is contrary to the homogenous distribution where connectivity is evenly distributed through the whole area.

A network being connected or partitioned is an important property of connectivity for routing design. All the mobility models mentioned above could lead to a partitioned network if the transmission range is small enough. But, different models lead to different types of partitioned connectivity. The random models lead to *sparse* networks, whereas these "heterogenous" models often lead to *clustered* networks. What distinguishes a clustered network from a sparse network is the number and size of its (connected) components. Due to this qualitative difference in component size, in sparse networks it is typically sufficient networks to restrict communication between nodes to single hops, as this does not sacrifice many relay opportunities. For this reason, many existing routing algorithms for partitioned networks exploit only one-hop communication. In contrast, in clustered networks, restricting communication to single hop relaying would be very limiting, given the larger clusters and the presence of longer paths.

Beside mobility models that describe the connectivity of the nodes indirectly, there have appeared recently models that define directly the connectivity of the nodes through a distribution of inter-contact and contact times for node pairs [CHD+07].

These models appear to be useful in the context of partitioned networks.

A more detailed survey of mobility models can be found in [CBD02], [FH04] and [MM09].

So, we see that there are many proposed mobility models. The question is which one to use. To answer this question we need to know what mobility of people in reality is like.

1.2 Real Mobility

We survey efforts to collect mobility data in settings of the possible realistic applications. We distinguish three types of collected mobility data sets: AP-based (Access Point based), contact-based, and position-based.

Most of the available data sets are AP-based. These data are collected in wireless LANs on university campuses or business premises. There is also one data set available from a metropolitan wireless network. In these data, the locations of nodes are shown to the granularity of access points (APs). APs in these networks are similar to base stations in conventional cellular networks. These data are more useful for the analysis of the usage of a network than for the analysis of the mobility of users. The reason is that the used devices are laptops or PDAs that are usually turned on when a node uses the network, otherwise they are turned off. Hence, these data captures just a small part of real mobility. Also, as mentioned above the granularity of locations is in scale with the APs.

However, there are some general conclusions about real mobility that the researchers found in these data-sets. First, there exists a node diversity where most of the nodes do not move much, but some nodes are highly mobile ([TB00],[BC03]). Second, a space diversity is also present, where users spend most of their time in their *home* locations ([BC03],[KE05]). Third, a heterogenous spatial distribution of users appears in many traces, i.e., there are some APs with on average a larger number of users than in others APs ([TB02],[SB04]).

Contact-based mobility data sets are those sets that give information for every node pair if the two nodes are connected ([SCP$^+$04], [CHD$^+$07], [LWM06]). Thus, there is no space information here. From this information, researchers characterize mobility through a distribution of contact and intercontact times of nodes' pairs (times that two nodes are connected and time between two contacts). These data are of relatively small scale (20 nodes in [SCP$^+$04] and [LWM06]), and 54 nodes in [CHD$^+$07]), so we did not find them satisfying enough for our analysis.

Ideal mobility data would be a large-scale position-based data that give us the geographic coordinates of nodes frequently enough (say about every 10s) [KWSB04]. When we started our investigation and when we looked at realistic mobility, there was only one position-based data set available. But, this data set consisted of only two users. From these data we could see an obvious clustering of the users' movements in specific locations, hence the space diversity was very prevalent.

Therefore, at the time we started our research there were no good enough mo-

bility data available through which we could validate mobility models. This is still a very open research area.

1.3 Routing Mechanisms

Depending on the application, the requirements of routing can be different. Many applications require a unicast routing service, where a source node s wants to deliver a message to a destination node d. Social applications and content spreading ones require a broadcast or multicast routing service, where a node wants to deliver a message to all nodes or to a group of nodes in the network. And, some applications require a more specific routing service, such as a geocast routing, where a node wants to deliver a message to nodes located in a specific area.

Furthermore, applications have a different requirements regarding performance metrics of routing. The most important performance metrics for the design of any routing protocol are: delivery rate, delay, and throughput. Delivery rate is the percentage of messages delivered to their destinations. Delay is the time from when a source sends a message until the destination receives it for the first time. Throughput represents how much data transfer in total can be sustained by a network.

What is desirable depends on the application, but what is achievable depends on the environment where a network operates. According to the delay, we distinguish an instantaneous and delay tolerant routing service. In the instantaneous service, we expect that a message is delivered in a relatively short time (of order of ms or s), whereas in the delay tolerant service, we can tolerate very large delays (of order of even days). The instantaneous service is possible only in environments where an assumption about an end-to-end connectivity holds. This assumption means that between any pair of nodes in the network there is a communication path (possible multi-hop through other nodes) at all times. Networks can be partitioned (i.e, the end-to-end connectivity assumption does not hold) either because of a nodes' mobility pattern or radio channel conditions. Networks where only delay tolerant service is possible are often called Delay Tolerant Networks (DTNs).

In this work, we consider the problem of providing an efficient unicast routing service into realistic challenging environments where the assumption about the end-to-end connectivity may not hold. Therefore, we next review existing mechanisms for the unicast routing in mobile ad-hoc networks. These mechanisms differ depending on the assumption about the end-to-end connectivity in a network. We first review routing algorithms for the end-to-end connected networks and then for the partitioned ones.

1.3.1 Connected Networks

Routing algorithms for connected ad hoc networks can be classified as proactive (or table driven), reactive (or on-demand), hybrid (or cluster-based), and position-based (or location-aware) ([BK07], [CDB$^+$07]).

Proactive algorithms are based on conventional link-state or distance-vector routing algorithms. They try to maintain shortest-path routes by using periodically updated views of the network topology. Two examples are DSDV and OLSR.

DSDV [PB94] is based on the classical Bellman-Ford algorithm. Every node has a routing table with the "next hop" for every reachable destination and the minimum distance (number of hops) to the destination. Whenever any change in the table happens, it is reported to neighboring nodes and thus the tables are updated.

OLSR [CJ03] is based on the link state routing. Every node maintains information about the network topology, which is then used to determine the shortest-path routes. Each node determines the link costs to its neighbors by broadcasting "hello" messages periodically. Whenever any change in the costs appears, the node broadcasts this information to all other nodes.

Reactive algorithms determine routes only when necessary, typically when a source needs to sends a packet and the source does not know any route to the destination. Two examples are DSR and AODV.

DSR [JM96b] uses a source routing, where a source node includes a route to be followed by a packet in the packet's header. The source discovers this route on-demand when the source does not know it. This is done by broadcasting a route "request" packet.

AODV [PBRD03] is an on-demand extension of DSDV. Like DSDV, every node has a routing table with the next hop and the minimum distance to each destination. However, the routes are created on-demand, that is only when there is no "fresh" entry in the table. AODV determines the freshness of information by maintaining the time that an entry was last utilized. An entry in the table is deleted after a certain threshold of time.

Hybrid algorithms are a combination of the reactive and proactive principles. For example, ZRP [Haa97] divides the network into zones or clusters of nodes. The nodes within a zone use a proactive algorithm. Every zone has a set of peripheral nodes. When a node send a packet to the destination for which it does not have an entry in its routing table, it requests the peripheral nodes to reactively discover the route. More advanced proposals are [KSH07], [WCYZ07].

Division of the network into zones or clusters is called clustering. There are many proposed algorithms for this problem [CLL04], [CR09].

Position-based algorithms use geographical positions of nodes in the network to make routing decisions. These positions can be obtained by using GPS receivers.

For example, GPSR [KK00] uses only neighbor locations in forwarding data packets. In its greedy forwarding scheme a node forwards a packet to a neighbor that is geographically closest to the destination. This procedure is repeated until the destination is reached. If the packet ends in a dead-lock, i.e., if all neighbors are farther from the destination than the node itself, then the node performs another procedure called a perimeter forwarding.

GPSR and similar algorithms (e.g., [BLBG05], [GS07]) require a location service in order to find out a position of the destination. This location service may in-

cur a large transmission cost. There are different proposals for eliminating the cost due to the location service , e.g., GLS [LJDC+00], GRSS [Hsi01], LER [GV06]), where an approximate location service is used.

1.3.2 Partitioned Networks

Routing mechanisms for connected networks described above cannot work in mobility environments where the assumption about the end-to-end connectivity does not hold. That is to say, if a destination of a packet is not in the same portion of a network as the packet's source then the packet would not be delivered to the destination but the packet would be dropped. This is because in the reactive and proactive algorithms it would not be possible to find a route towards the packet's destination. And, in the position-based algorithms the packet would end in a deadlock.

A different mechanism, called *store-carry-forward* or *mobility-assisted forwarding*, must be used in such networks. In this mechanism, a node needs to store and carry a message with itself, until a better opportunity to forward the message arises. Thus, nodes use their own mobility to move a message towards the destination and this could be the only possible way to deliver a message to the destination. Finding such routes through space and time is obviously a complex problem in general and depends heavily on the joint statistics of link availability [JFP04]. The delay incurred may be large and thus in such networks only the delay tolerant routing service is possible.

One of the simplest possible store-carry-forward mechanisms is a *direct* forwarding, where a source forwards a packet only to the packet's destination. Thus, the source stores and carries a packet until it meets the destination, and then it forwards the packet to the destination. This algorithm has the minimum possible transmission cost, i.e., only one transmission per message. But the delay is large, and the delivery probability may be low, depending on the mobility pattern.

Another simple mechanism is an *epidemic* routing (ER) [VB00]. Here, a source node gives a copy of a message to every node it meets (that did not already receive the message). Also, nodes with a copy of the message disseminate the message further to other nodes they meet. This algorithm has the minimum possible delay and the maximum possible delivery probability (under the assumption of infinite buffers), but it incurs a very large transmission overhead (in the network of n nodes ER incurres $(n-1)$ transmissions per message).

There are proposed mechanisms in between these two. They are all heuristic-based mechanisms that try to achieve a better trade-off between delay and throughput. They try to limit flooding of ER, but still to have delay close to ER and also a good delivery probability. Next, we review these mechanisms.

First, we could limit a number of hops a message can traverse [VB00] in ER. Second, we could add an ack mechanism in ER where a destination sends an ack when it receives a message. This ack is then broadcasted in order to delete other copies in the network [VB00], [HABR05]. Third, we could limit the time that a

message lives (i.e., we could add a time-to-live mechanism) [HABR05].

Fourth, an algorithm Spray and Wait [SPR05] limits a total number of copies made. Initially, when a message is originated, a number L of copies are sprayed, and then each of these copies is further routed only by means of the direct forwarding. There could be different mechanisms of the initial spraying. The authors in [SPR05] consider two: a source and a binary spraying. In the source spraying, a message's source forwards all L copies by itself to the first L distinct nodes it meets. In the binary spraying the source of a message initially starts with L copies (tokens), and then any node that has $n > 1$ copies hands over $\lfloor n/2 \rfloor$ to another node it meets (with no copies) and keeps $\lceil n/2 \rceil$ for itself. It is shown that this binary spraying mechanism is the optimal one.

Fifth, there are proposals to make more clever routing decisions based on the mobility history of nodes. We call these mobility-based approaches. The idea is to copy a message only to the nodes that are more likely to meet the destination in the future or to meet nodes that will meet the destination. This probability is determined based on a mobility history. In PROPHET [LDS03b], this probability is calculated based on how recently and how frequently a node meets the destination or nodes that meet the destination frequently or recently. Whenever two nodes meet, they exchange and update their delivery predictability, which are then used to decide whether to exchange messages or not.

Sixth, there are *single-copy* proposals, contrary to the multi-copy ones described above. In the single-copy algorithms, a node does not copy a message to other nodes, rather it forwards a message to a better node. Thus, only one copy of a message exists at all times. The advantage of forwarding and not copying is a savings in the buffer space as well as a smaller delay or a larger delivery probability.

Hence, in these single-copy algorithms, when a node with a message meets another node, then that node decides whether to forward the message or not, by using a mobility-based approach similarly as in PROPHET. A node is suitable for forwarding a message if the delay of a message is smaller and/or the delivery probability is larger if the message is forwarded to that node. There are different proposals of how to make decisions about whether to forward the message or not.

Thus, in Shortest Path Routing [TZZ03] a basis of the decisions is an estimation of the probability that two nodes will meet. This probability is calculated as a time that two nodes stay connected dividing by an observing time window, i.e., $p_{ij} = Time_{connection}/Time_{window}$ is a probability that node i meets node j. In Practical Routing [JLS07] a basis of the decisions is an estimation of an expected delay for a message to go from one node to another. The calculated metric is $\sum_{i=1}^{n} d_i^2/2t$, where d_i is the duration of a disconnected period i, and t is a total observed period. Then, in both of these algorithms when two nodes meet they update these metrics. Moreover, a network topology graph is maintained. This graph shows the described metrics between all nodes in the network. Then, the forwarding decisions are made by finding the shortest path routes in this graph.

In a MobySpace algorithm [LFC07], the idea is that two nodes are more likely to meet each other if they have similar mobility patterns. Hence, they propose

to use the formalism of an Euclidean virtual space (called MobySpace) as a tool to make forwarding decisions. Routing is done by forwarding messages towards nodes that are increasingly similar to the mobility pattern of the destination. [CM01, MHM05] Seventh, DTC [CM01] and CAR [MHM05] use a *multi-hop* mechanism that differs from all others mentioned. They do not consider only one-hop neighbors as candidates to forward a message, but also all reachable multi-hop neighbors. Also, they differ in that a node with a message periodically makes routing decisions at every *rediscovery* interval, and not upon an encounter of a new one-hop neighbor as in other algorithms. Beside these differences, they use a similar mobility-based single-copy approach as others.

DTC proposes to use also some other metrics (the authors call them utilities) besides these mobility-based metrics. Thus, DTC uses mobility-based metrics (last encounter timers, frequencies of encounters and future plans) and other metrics (a nodes' power and the rediscovery interval). CAR generalizes this concept of using utilities by using a general mathematical framework for the evaluation and prediction of these utilities.

Eight, BUBBLE [HCY08] uses social structures (communities) in human mobility to make routing decisions. In particular, BUBBLE uses heterogeneity in human interaction, both in terms of hubs (popular individuals) and groups or communities.

Ninth, network coding and erasure coding techniques are also proposed for routing in DTNs. In both of these works they show that this could improve network throughput. In network coding [WLB05] intermediate nodes can combine, instead of simply forwarding, packets received so far, and send them as new packets. In erasure coding [WJMF05] a message of k blocks is encoded into $n > k$ blocks in such a way that if k or more of the n blocks are received, the message can be successfully decoded. In both ways, network throughput is improved.

In some scenarios of DTNs, important considerations are a limited transfer duration, transfer rate and storage space. As a consequence of this, an important problem is buffer management (i.e., what to do when a buffer is full and a new message comes and how to order messages to be transferred upon a transfer opportunity). There are several proposals for the buffer management [DFL01, BBL05, RHB$^+$07, BLV07, WHAB09], and this is still an active area of research. In the design of our routing algorithm, we do not explicitly address these limitations and we assume that transfer duration, transfer rate and storage space are not scarce resources. Thus, the buffer management is out of the scope of our work. But note that we do take implicitly these into account, by designing a scheme with small transmission overhead in total and with a mechanism for discarding unnecessary copies of messages.

1.4 Contribution

In this thesis, we endeavor to identify features of mobility common to different applications, in order to devise routing methods that are tailored to exploit these features. We explore two features in particular, (i) predictability and (ii) stable and heterogeneous spatial node distribution.

1.4.1 Exploiting Predictability

A mobility process is predictable if the future location of a node can be well estimated given knowledge of its current and past locations, and possibly other statistics. We show how the performance of routing can be improved by explicitly incorporating mobility prediction. Specifically, we consider the performance of Last Encounter Routing (LER) under a simple synthetic random waypoint (RWP) mobility model. We extend the LER algorithm so that it takes into account predicted node trajectories when making routing decisions, and we show that this significantly improves its performance.

Last Encounter Routing (LER) [GV03, DFGV03a, DFGV03b] refers to a type of routing algorithm where the destination of a packet is located without the help of a location service and without any control traffic to track topology changes due to node mobility. Rather, a packet is routed using only the encounter histories at nodes it is forwarded through. In its basic form, the encounter history consists of the time and location when a node was directly connected to another node.

It is clear that the performance of LER algorithms is closely tied to the mobility pattern of the nodes in the network. To see this, consider an extreme scenario where there is no dependence between a node's position at different times, i.e., nodes "jump around" randomly in the network domain. In this case, history information is of no use, and any LER algorithm would perform as poorly as an exhaustive search.

We look at the RWP mobility model in the context of LER for several reasons. First, the model is well studied and is very prominent in simulation studies of mobile ad hoc networks. Second, as shown in [GV03], a good performance for the RWP model is harder to achieve than for another prominent mobility model, the random walk. Third, in contrast to the random walk, the RWP model is predictable. This provides us with an opportunity to exploit additional information collected in an encounter (such as speed, direction, etc.) to improve routing.

We compute optimal predictors for the RWP model under different observation information by using the minimum mean square error as the criterion. Then, we incorporate these predictors into GREASE, an instance of LER algorithms [GV03], by making prediction of nodes' locations in the encounter history. We show through simulations that this GREASE-RWP algorithm achieves a drastically better performance than the "non-RWP" version reported in [GV03]. Specifically, the total average route cost is slightly more than twice as many hops as the shortest path. This is quite remarkable, given that no resources were invested to track the

rapid change in the network topology due to RWP mobility. Moreover, the cost of GREASE-RWP routes, relative to the shortest path, does not seem to increase as we scale up the network size.

Predictability is a mobility feature recognized in many realistic scenarios. For example, the limitation of the speed of nodes makes a mobility process predictable over at least a short time scale. Another example is that nodes may have some locations that they visit more often than others, as mentioned in Section 1.2. Hence, the predictability is exploited by many routing mechanisms designed for the partitioned networks, as we see in Section 1.3. The novelty of our work is that we consider the prediction in the RWP model, and how to incorporate this prediction into LER, both of which have not been previously considered.

1.4.2 Stable and Heterogenous Spatial Node Distribution

The next step we perform in our thesis is to study realistic mobility data-sets in order to possibly identify other features common in many applications that could be exploited to help routing. As mentioned in Section 1.2, at the time we began this investigation, there was no suitable large-scale mobility data-sets with enough frequent location updates. Hence, we made a special effort to find such suitable data-sets. The results of this effort were two distinct large-scale data-sets of taxis in the city of Warsaw, Poland and in the city of San Francisco, USA [1]. The Warsaw data-set consists of ~ 800 taxis over a three-month period, and the San Francisco data-set consists of ~ 600 taxis over a month period.

We distinguish the following properties:

- spatial distribution of nodes is heterogenous rather than homogenous
- there exist regions of the dense connectivity, which we call concentration points (CPs); these are regions where the node density is much higher than average, and where nodes have therefore a much better chance than on average of being connected to other nodes;
- a network is often partitioned with heterogenous connectivity, where the highly connected CPs are interspersed with low-connectivity regions;
- spatial node distribution typically remains stable over time; this is because it is determined by natural or constructed environment, which change over relatively long time-scale.

Although heterogenous spatial node distribution and existence of CPs has been observed by others (as mentioned in Section 1.2), their stability over time appears to be a novel one.

[1] The credit for the collection as well as data mining of these data-sets goes to Michal Piorkowski[Pio09]

Modeling

We model a network that possesses these observed properties with a set of stable concentration points (CPs) characterized by high node density, with a mobility process that describes how nodes move between these islands of connectivity. We study two instances of this model: the G-model and the H-model.

In the G model, we view the network as a *mobility* graph $G(V, E)$, where the vertex set V represents the CPs, and the edge set E represents flows of mobile nodes between the CPs. Two nodes can communicate with each other only if they are at the same vertex; and if a node is at an edge, then it can not communicate with any other node. This assumption of a "sharp" connectivity helps us to first concentrate on an essence of the problem by abstracting away all other complexities of a real mobility.

In the H-model, contrary to the G-model, nodes move in a geographic space with two nodes being connected if they are within a certain transmission range. Thus, the H-model relaxes the sharp connectivity assumption. In the H-model, every node performs an independent random walk on a plane, with heterogenous speeds in different areas of the plane. Specifically, in the plane we randomly place disks that represent CPs. A node moves faster in the area of the disks than outside of the disks. Because of this difference in the speeds in different regions, the node density is higher in the CPs than outside of the CPs. The H-model therefore results in a high connectivity inside the CPs and a low connectivity outside of them.

Compared to the other existing mobility models that could lead to a temporally stable and spatially heterogenous node density with an appropriate set-up of their parameters, we find the H-model more interesting for the following reasons. First, the H-model is parsimonious, requiring only a small number of parameters to control the macroscopic properties of the model. We avoid over-specialization in the model design and we propose a generic mobility model, that can be further closely tuned to a specific real-life mobility scenario. Nevertheless, the model captures clustering and spatial heterogeneity. It is easier than in other existing models to tune parameters properly in order to get the desired spatial heterogeneity. Moreover, it is easy to tune the parameters to set-up the desired nodes' density in a stationary regime inside and outside of the clusters.

Second, another interesting macroscopic property of the H-model is the dynamics of connectivity, i.e., the stability of cluster locations. As we will see later in our work, for DTN routing protocols, it is not only an instantaneous appearance of clusters that matters; how these clusters evolve over time is also important. In reality, cluster formation can often be attributed to features of the natural or constructed environment (e.g., railway station, warehouse, parking lot, watering hole). For DTN routing, this is an important feature that can be exploited for designing efficient schemes.

Third, although other models rely on similar mobility processes, e.g., random waypoint or random direction, they assume that each node has a set of so-called *preferred locations*. Such locations are visited by nodes more often than other lo-

cations. In other words, nodes are not statistically equivalent, and their individual mobility can be well predicted. This is a strong assumption, with important implications for the design of robust routing protocols. This is not the case in the H mobility model, where nodes move independently following the same law. We also provide evidence that in some realistic scenarios, the mobility patterns of different nodes can be almost indistinguishable. This may be problematic for DTN routing protocols that assume a-priori that the movement patterns of nodes are very different and predictable.

Forth, there is also one interesting aspect of the H-model that differs from other models. The H-model captures an interesting relationship between speed and density. In the model, differences in node density (and hence connectivity) in different regions can be viewed as a consequence of different average speeds in these regions. We observe such an inverse relationship between speed and density in at least one realistic mobility scenario, but we suspect that it may be quite universal.

1.4.3 Island Hopping through Stable Concentration Points

We develop a routing algorithm assuming the mobility model described in the previous subsection. Our Island Hopping (IH) algorithm is a novel approach to routing for mobile partitioned networks; it explicitly exploits the assumed stable CP topology in the nodes' mobility.

More specifically, nodes make routing decisions for a message using:

- a CP graph $\widehat{G}(V, E)$, a node's representation of the CP topology with V representing the CPs and E representing possible flow of nodes between the CPs;
- the nodes' locations in the CP topology;
- a location of the message's destination node of the message.

Based on this, nodes decide on a sequence of CPs through which the message is forwarded to its destination, rather than deciding to which nodes to give a message as in other compared algorithms.

The key question is how to pass a message from one CP to the next CP through nodes whose future movements are random and unpredictable. If the future movements of nodes were known, we could pass the message to a single node that would move in the right direction, i.e., to a CP closer to the destination in $\widehat{G}(V, E)$. However, given that future movements are unpredictable, our algorithm makes a small number of copies of a message at each CP, in the hope that at least one copy will move to the intended next CP and the other copies will be discarded. The process repeats at the next CP, until the message reaches its destination.

Therefore, we view our algorithm as a spraying of a small number of copies at CPs along the shortest path in the CP topology graph. In this way our algorithm achieves a very good delay-throughput trade-off , i.e., it achieves delays of the order

of much more aggressive flooding-based schemes and requires a much smaller number of copies of each message.

We compare the performance of our algorithm through simulations with Direct Forwarding, ER, PROPHET, SW, and SF. We assume infinite buffers in all algorithms. We see that our protocol achieves a delay of the order of ER and PROPHET (which is much less than in the direct forwarding), while its transmission cost is much smaller (in some scenarios about ten times less). Compared with SW and SF, our IH achieves about 35% less delay for the same transmission cost. Moreover, the scalability with the network size appears to be more favorable for IH than other compared algorithms. The delivery probability is similar in all algorithms.

Our IH algorithm differs in several aspects from other proposed approaches of routing in partitioned networks. First, an assumed mobility model is explicitly different than other models. A distinguishable property of our model is the clustering and the stability of cluster locations. In reality, cluster formation can often be attributed to features of the natural or constructed environment (e.g., railway station, warehouse, parking lot, watering hole). For DTN routing, this is an important feature that can be exploited to design efficient schemes. Although other proposed models [HMS+05, LDS03a, LYD06, HSPH07] can lead to the clustered networks, they differ in the following. They assume that each node has a set of so-called *preferred locations*. Such locations are visited by a node more often than other locations. In other words, nodes are not statistically equivalent, and their individual mobility can be well predicted. This is a strong assumption, with important implications for the design of robust routing protocols. This is not the case in the G and H mobility models, where nodes move independently following the same law. This may be problematic for DTN routing protocols that assume a-priori that the movement patterns of nodes are very different and predictable, e.g., [LDS03b, LFC07]. Furthermore, in the context of DTN routing, the mobility is usually modeled by considering pairwise contact duration and inter-contact times. In contrast, we argue that it is also important to consider the collective mobility characteristics and resulting patterns that may not be captured by pairwise statistics. Second, most of other routing proposals [VB00, SPR05, LDS03b, TZZ03, JLS07, LFC07], with the exception of [CM01, MHM05], exploit only the one-hop communication between directly connected neighboring nodes. Our algorithm exploits in addition a multi-hop communication through a connected component, both for forwarding/copying messages and for exchange of control information. This is important for learning the collective mobility patterns that are then exploited to make efficient routing decisions. In this respect, the closest approach to ours is the social-based routing algorithm BUBBLE [HCY08]. It collectively learns a network clustered structure that arises due to social human interactions. The deference is that in our model there is no preference in meetings between specific nodes, rather all nodes pairs are statistically the same. Hence, under our model it would be impossible for BUBBLE to find social communities and to make efficient routing decisions.

1.4.4 COllaborative GRAph Discovery (COGRAD)

In many applications, it is unrealistic to assume that the graph of CPs and the flows of mobile nodes between CPs is known a-priori (though this could be the case in some applications). Instead, we assume that the only information that nodes have available is the set of other nodes that they can reach (either directly or over multiple hops), which can be discovered in a straightforward manner (hello messages, flooding, etc). Therefore, a large part of this thesis is dedicated to devising a distributed algorithm that allows the nodes to collaboratively discover the CP graph, *in the absence of any signal from the environment*, such as GPS coordinates or fixed beacons. This problem is a novel one, i.e., to the best of our knowledge there is nothing similar in the research literature. We call a class of algorithms for this problem by COllaborative GRAph Discovery (COGRAD). Our COGRAD algorithm divides the problem into two phases: vertex labeling and edge discovery.

Vertex Labeling. The goal of this phase is to generate a label, i.e., a unique identifier for each vertex of V, which will remain stable over time, even though nodes move in and out of each vertex. Suppose that at a given time the nodes currently located at the same CP agree on a label for this vertex. Now another node i arrives at this vertex. Node i has not received any explicit clue from the environment that it has moved, and the other nodes have not received a clue that they have not moved. However, node i's set of neighbors has changed rather markedly, whereas the other nodes' neighbor set has only seen the addition of i. These nodes can therefore decide jointly that it is likely that node i has moved, and the other nodes have not; node i therefore accepts the label of this vertex.

Edge Discovery. Once we have associated a label with each CP, a node can discover the edges of the CP graph as it moves from one CP to the other CP. To ensure that each node learns the entire graph, even though it may only visit part of the graph, it is necessary that nodes exchange edges they have discovered. This also accelerates the learning process of the nodes, and moreover this allows for outdated information (e.g., a label that does not exist any more) to be flushed out.
We first develop a COGRAD algorithm under the G-model. We show that it successfully replaces a COGRAD "oracle" in IH under the G-model. As mentioned in the previous subsection, the G-model makes the sharp connectivity assumption (i.e., the nodes at the same vertex are connected to each other, whereas a node at an edge is not connected to any other node). In addition to this, we make one more assumption in the design of COGRAD in the G-model. We assume that a node sees momentarily any change in its neighborhood. This implies that at one time instant only one node can leave or join a vertex. These assumptions make the design problem of COGRAD easier. Though in many applications unrealistic, they were helpful to make first steps towards the design of COGRAD under more realistic assumptions. Moreover, having a simple COGRAD scheme made it also easier to focus on the design of the main principles of IH.

Then, we develop a COGRAD algorithm under the H-model. This model relaxes the sharp connectivity assumption, i.e., nodes move in a geographic space with two nodes connected if they are within a certain distance of each other. We also relax the assumption about the continuous neighborhood discovery. Instead, we assume that a node sees changes in its neighborhood periodically every δt, where this could be implemented simply through physical-layer broadcast or through flooding algorithms. These relaxed assumptions led us to a different design of the vertex labeling algorithm, whereas the edge discovery part can stay the same. Integration of this COGRAD into the IH algorithm designed for the G-model requires minor modifications in IH in order for IH to successfully operate under the H-model.

How successfully the COGRAD algorithm operates, for both the G-model and the H-model, depends on control parameters of the algorithms. For the G-model we set-up the parameters by using our knowledge about the model, hence COGRAD for the G-model is model-dependent. For the H-model, we estimate the parameters in order to obtain model-independent COGRAD. Therefore, we design an adaptive COGRAD algorithm where nodes estimate the necessary parameters by themselves in a distributed way adaptively over time. We show through simulations that IH with adaptive COGRAD operates successfully. Although the self-adaptive COGRAD is designed for the H-model, we believe that it is also applicable in other mobility models that have the observed property of a stable CP topology.

Through evaluation of the adaptive COGRAD, we find that in some cases it performs poorly, giving unstable labels. This happens, for example, when two CPs are too close to each other and if they become disconnected and reconnected often. The main reason behind this is that the designed COGRAD algorithms are component-based, associating a label with a component, i.e., there can be only one label per a component. Therefore, we design a new *soft* labeling algorithm that allows several labels to coexist in the same component. We show that the soft labeling algorithm performs in a wider range of the H-model parameters than the component-based algorithm does. For the purpose of the comparison, we use a metric that shows how close a labeling algorithm is close to the idealistic one (i.e., when each CP has only one unique label). Simulation results show clearly that the soft labeling approach is a promising one for making labeling more robust to mobility conditions possible in reality.

1.4.5 Summary of Contributions

The main contributions of this dissertation are:

- An improved LER algorithm under the RWP model: exploiting the predictability of the model.
- Observations of common properties of realistic mobility processes: temporally stable and spatially heterogenous node distribution that leads to the partitioned connectivity with the stable CP topology.

- Two novel mobility models that possess the above mentioned properties: the G-model and the H-model.
- Island Hopping (IH): a novel mobility-assisted routing algorithm for networks with the observed properties.
- COllaborative GRAph Discovery (COGRAD): a distributed algorithm for discovering the CP topology.
- Evaluation: We evaluated IH both under an assumption that there is an oracle that reveals the CP graph to nodes and under our COGRAD algorithm in our two mobility models. We compared it with several DTN routing algorithms and we have shown a very good delay-throughput trade-off comparing to others.

1.5 Dissertation Overview

This dissertation is organized as follows. In Chapter 2, we show how the performance of LER under the RWP model can be improved by explicitly incorporating mobility prediction. In Chapter 3, we show how we model realistic mobility with the G and H mobility models. In Chapters 4 and 5, we design and evaluate our routing algorithm, Island Hopping, under the simplifying G-model and the more realistic H-model, respectively. Chapter 6 concludes the dissertation.

Chapter 2

Last Encounter Routing under Random Waypoint Mobility

In this chapter we show how the performance of routing in mobile ad hoc networks can be improved by explicitly incorporating mobility prediction. Specifically, we consider the performance of Last Encounter Routing (LER) under a simple synthetic random waypoint (RWP) mobility model. We extend the LER algorithm so that it takes into account predicted node trajectories when making routing decisions, and we show that this significantly improves its performance.

Last Encounter Routing (LER) [GV03, GV06, DFGV03a, DFGV03b] refers to a type of routing algorithm where the destination of a packet is located without the help of a location service, and without any control traffic to track topology changes due to node mobility. Rather, a packet is routed using only the encounter histories at nodes it is forwarded through. In its basic form, the encounter history consists of the time and location when a node was a directly connected neighbor of another node.

In [GV03], the following model was considered. A set of nodes perform independent random walks on a square lattice. Two nodes are directly connected neighbors if they reside at the same lattice point. Every node remembers when and where it has encountered every other node, in a *last encounter table*. A very simple algorithm called EASE was introduced in [GV03] to compute a route from a source node to a destination node, based only on LE history at every node. It was shown that the expected total cost of EASE routes is a small multiple of the expected shortest path length between a random source and destination. In other words, EASE is a scalable LER algorithm for the random walk mobility model, as the cost of routes relative to the shortest path does not blow up as the network size increases.

It is clear that the performance of LER algorithms is closely tied to the mobility pattern of the nodes in the network. To see this, consider an extreme scenario where there is no dependence between a node's position at different times, i.e., nodes "jump around" randomly in the network domain. In this case, history information

is of no use, and any LER algorithm would perform as poorly as an exhaustive search.

An important question, then, is the interplay between the mobility model and the performance of LER algorithms. A complete answer to this question remains elusive, but it is possible to develop some intuition about features of the mobility model that are favorable to LER algorithms. In particular, we argue that the following three features of a mobility model help LER: locality, frequent intersections, and homogeneity. *Locality* means that a node's position at a time t correlates with its position at a certain time in the future; this ensures that information about past encounters is actually useful in locating a node. *Frequent intersections* mean that a node over a given time interval tends to encounter a large number of other nodes. This ensures that information about that node's location is refreshed frequently. *Homogeneity* means that the statistical properties of each node mobility are similar. This ensures that the speed of diffusion of encounter histories due to movements of other nodes is matched to that of a destination node. In this paper, all the scenarios we consider are homogeneous.

Based on these three features, we can see that the random walk mobility model is quite advantageous to LER. First, as a node performs independent steps over time, the dependence between the current and a future position of the node decreases only slowly with time. Specifically, it follows from the central limit theorem that the difference between the two positions t seconds apart is a random variable with variance proportional to $\sqrt{t}$. Second, despite the locality, a node encounters other nodes frequently. Specifically, it was shown in [GV03] that over a time interval of length t, a node encounters $\Theta(t/\log t)$ other nodes. As a result, in the random walk mobility model, we observe that (i) a node's past location contains information about its current location over a relaxation time of $\Theta(n)$, where n is the network size, and (ii) when a node moves by a distance d, it encounters approximately $\Theta(d^2/\log d)$ other nodes. This means that information about a destination node's movement is quite dense around this node, which helps LER algorithms, and that such an algorithm can rely on fairly old information to locate a destination.

In this chapter, we consider LER under the random waypoint (RWP) model. There are several reasons why we are interested in LER for the RWP model. First, the model is well studied and is very prominent in simulation studies of mobile ad hoc networks. Second, it was shown in [GV03] that the LER algorithms EASE and GREASE did not perform well with this model. Third, given the discussion of helpful features above, the RWP model is much less favorable to LER than the random walk.

To see this, let us perform a back-of-the-envelope comparison with the random walk. The random waypoint model is is defined (informally) as follows. Every node moves independently of any other node. A node selects a random waypoint uniformly in the area of the network, and moves towards this waypoint in a straight line and at constant speed. Once it reaches this waypoint, it selects a new waypoint independently of the previous one, and starts moving towards it (possibly

after some pause time), and so forth. So, in this model the average transition time between waypoints is $\Theta(\sqrt{n})$, proportional to the diameter of the network. Once a node has gone through a waypoint, the dependence with its current position drops very rapidly. A past position of a node is therefore only useful for $\Theta(\sqrt{n})$ time. Furthermore, when a node moves by a distance d, it encounters only $\Theta(d)$ other nodes. This means that information about a destination node's movement is much less dense around this node, and that an LER algorithm must rely on fairly recent information to catch up with a destination.

On the upside, the RWP model has a feature that is in our favor: the node movement is highly predictable over a short time-scale, because a node moves at constant speed on a straight line between waypoints. This provides us with an opportunity to route a packet towards the predicted current position of its destination, rather than simply towards the location of the encounter with the destination. This is not possible in the random walk case, where the best predictor for a node's future position is simply its current position.

We devise a version of the GREASE algorithm (in [GV03]) that is tuned to the specific features of the RWP mobility model. GREASE-RWP takes into account the short relaxation time by reducing the encounter age it searches for initially to obtain a first estimate of the destination's location. It then uses mobility prediction as the packet approaches the destination in order to move towards the destination in "shortcuts". We evaluate GREASE-RWP through simulation in networks with up to 1000 nodes, and we find that GREASE-RWP has a significantly better performance than the original GREASE, and seems to scale with network size.

The chapter is structured as follows. In Section 2.1, we formally define the RWP model and discuss some of its properties that matter in the context of LER. In Section 2.2, we present a LER algorithm specifically designed for the RWP model. We present simulation results in Section 2.3. Section 2.4 concludes the chapter.

2.1 The Random Waypoint Mobility Model

In this section we formally define the RWP model, and discuss the properties that play an important role in the performance of LER.

Nodes move on a square torus of side a. The origin is in the center of the square, and the axes are parallel to the sides of the square. The vector $c - b$ is the vector on the torus surface such that $\|c-b\| \equiv d(b,c)$ is the shortest distance between the points b and c. Nodes move independently of each other, so it is enough to define the movement of one node.

The RWP model, with constant speed and no pause time, in the torus of side a, is completely described by:

- the value of the node's speed v
- the sequence of independently and identically distributed (i.i.d.) random

variables $\{P_w\}_{w\in\mathbb{N}}$, uniformly distributed in the torus.

The P_w is the w-th waypoint. If the node moves between the w-th and $w+1$-st waypoints we say that it is on the w-th segment. The distance between these waypoints $L_w = d(P_w, P_{w+1})$ is the w-th segment length. The time for the node to traverse the w-th segment is the w-th segment duration. The time instant S_w when the node is at the w-th waypoint is equal to:

$$S_w = \sum_{j=1}^{w-1} \frac{L_j}{v},$$

where $S_1 = 0$. The speed vector between the w-th and $w+1$-st waypoints is equal to:

$$V_w = v\frac{P_{w+1} - P_w}{L_w}.$$

Therefore, the node's position at time t can be formally expressed as:

$$X(t) = P_{w(t)} + V_{w(t)}(t - S_{w(t)}),$$

where $w(t)$ is an index such that $t \in [S_{w(t)}, S_{w(t)+1})$.

Note that the L_ws are i.i.d, thus the random process $\{S_w\}_{w\in\mathbb{N}}$ is a renewal process. This property comes from the fact that torus represents isotropic space where all points are equivalent.

In the remainder of this section, we assume that time $t = 0$ is the beginning of the observation period for the renewal process $\{S_w\}_{w\in\mathbb{N}}$ that has been operating long enough to be in steady state[1]. We enumerate the points of the renewal process as $\{S_w\}_{w\in\mathbb{Z}}$ with the convention that $S_0 \leq 0 < S_1$.

2.1.1 Relaxation Time

The key property of realistic mobility processes that LER exploits is that the location of a node at a time t and a time $t+\tau$ are dependent, and that therefore information collected through an encounter at time t can be useful for a packet looking for its destination at time $t+\tau$.

More specifically, consider a realistic mobility scenario where nodes have limited speed. Suppose that a source knows the destination position $X(t)$ and wants to route a packet to the destination at time $t+\tau$. If τ is relatively small, then the destination is in the small area around $X(t)$. Thus, information $X(t)$ is useful. If τ increases, then the area where the destination may be becomes larger and information $X(t)$ is less useful. For some large τ, the destination may be anywhere in the network area and information $X(t)$ is useless.

The relaxation time T_r is the minimum τ for which $X(t+\tau)$ does not depend on $X(t)$ any more (note that as we assume homogeneity, all nodes have the same

[1]The RWP model with random speed does not possess a steady state in terms of average speed [YLN03]. Here, this problem does not arise because of constant speed.

relaxation time). The relaxation time is the maximum time after its observation that past information about a node's movement is still of use. It is therefore crucial to ensure that a LER algorithm does not rely on observations older than T_r, as this would lead to poor routes.

Let us consider the relaxation time in the RWP model. Let $Y_t = S_{w(t)+1} - t$ be the time the node travels to the next waypoint. After time $Y_t + L/v$ (L denotes the distribution of the segment lengths $L_w s$) the node will reach the second next waypoint. Hence, $X(t+\tau)$ for $\tau > Y_t + L/v$ depends only on P_ws, where $w \geq w(t) + 2$. The $X(t)$ depends only on $P_{w(t)}$ and $P_{w(t)+1}$. Therefore, the positions $X(t)$ and $X(t+\tau)$ are independent for $\tau > Y_t + L/v$.

2.1.2 Prediction

In the RWP model, a node moves in long straight lines. This results in fewer encounters with other nodes relative to the traveled distance, which is unfavorable to LER. But, this also implies predictability of a node's position in the future. Mobility prediction means prediction of a node's future position given the node's current position and additional observations about the node's mobility (e.g., the current speed, direction, ...). Our goal is to use prediction of the destination position in order to decrease the routing cost of LER.

The observations available to a node on its own mobility process depend on the scenario. At one extreme, a node might be able to determine only some basic parameters about its instantaneous movement, such as its direction and speed. This might arise when a node's movement is subject to external influences, and if the node lacks the capability to identify waypoints. At the other extreme, a node might be able to know its precise movements a long time into the future. This might arise when a node's movement is predetermined by the node itself (e.g., a person running errands in a city, or a doctor visiting patients). Obviously, the predictability improves as the observation set increases. As our focus here is on the "tough" cases, we only consider observation sets that do not look beyond the next waypoint. Specifically, we calculate mobility predictors for the following three different observation sets θ:

1. the node's current position and speed vector;
2. the node's current position, speed vector and previous waypoint;
3. the node's current position, speed vector and next waypoint.

The optimal mobility predictor is the one that minimizes in some sense the error between the node's predicted and true positions. We use the minimum mean square error criterion.

Definition 2.1 *The optimal mobility predictor of the node's position $X(t+\tau)$, given the set of observations $\theta(t)$ is the value $\widehat{X}(t+\tau)$ that minimize:*

$$E[(X(t+\tau) - \widehat{X}(t+\tau))^2 \mid \theta(t)]. \tag{2.1}$$

Our predictors are optimal under the following assumptions. First, we assume that we make the observation at a random time t. Second, we neglect the torus structure of the simulation area. Instead we consider Euclidian distance by representing the torus with a square in $\mathbb{R}^2$ of the side length a with the origin is in the center of the square. The origin is fixed at an arbitrary point of the torus. We justify this assumption by the fact that our goal is to find local predictors within a short time-scale (more precisely within one segment length). Hence, the distance traveled by a node during this short time scale becomes in most cases equal to the Euclidian distance.

Theorem 2.1 *The optimal mobility predictor of $X(t+\tau)$ in the defined RWP model if we know the following observations:*

- *the node's current position $X(t) = x(t)$*
- *the node's current speed vector $V(t) = v(t)$*

is given by:

$$\widehat{x}(t+\tau) = x(t) + \frac{v(t)}{v}(P(Z \geq v\tau)v\tau + E[Z1_{\{Z<v\tau\}}]), \tag{2.2}$$

where Z is a random variable with the pdf equal to:

$$f_Z(z) = \begin{cases} c(1 - \frac{z^2\pi}{a^2}) & \text{if } 0 \leq z \leq \frac{a}{2} \\ c(1 - \frac{z^2\pi}{a^2} + \frac{4z^2}{a^2}\arccos\frac{a}{2z} - \frac{2}{a}\sqrt{z^2 - \frac{a^2}{4}}) & \text{if } \frac{a}{2} \leq z \leq \frac{a\sqrt{2}}{2} \\ 0 & \text{otherwise} \end{cases}, \tag{2.3}$$

where $c = 6(a\sqrt{2} + a\ln(1+\sqrt{2}))^{-1}$.

Theorem 2.2 *The optimal mobility predictor of $X(t+\tau)$ in the defined RWP model if we know the following observations:*

- *the node's current position $X(t) = x(t)$*
- *the node's current speed vector $V(t) = v(t)$*
- *the node's previous waypoint $P_{w(t)} = p$*

is given by:

$$\widehat{x}(t+\tau) = x(t) + \frac{v(t)}{v}(P(Z \geq v\tau)v\tau + E[Z1_{\{Z<v\tau\}}]), \tag{2.4}$$

where Z is a random variable with the pdf equal to:

$$f_Z(z) = \frac{f_L(z + l_0)}{1 - F_L(l_0)}, \tag{2.5}$$

where $l_0 = d(p, x(t))$, and $f_L(l)$ and $F_L(l)$ are the pdf and the cdf of the segment length (see Lemma 2.1 in the Appendix).

Theorem 2.3 *The optimal mobility predictor of $X(t+\tau)$ in the defined RWP model if we know the following observations:*

- *the node's current position $X(t) = x(t)$*
- *the node's current speed vector $V(t) = v(t)$*
- *the node's next waypoint $P_{w(t)+1} = p_{next}$*

is given by:

$$\widehat{x}(t+\tau) = \begin{cases} x(t) + v(t)\tau & \text{if } v\tau \leq d(x(t), p_{next}) \\ p_{next} & \text{if } v\tau > d(x(t), p_{next}) \end{cases} . \quad (2.6)$$

Next, we give main ideas how we calculate these mobility predictors. The details can be found in the Appendix.

According to the well known result from statistics [GD04], the value $\widehat{X}(t+\tau)$ that minimizes (2.1) is equal to:

$$\widehat{X}(t+\tau) = E[X(t+\tau) \mid \theta(t)]. \quad (2.7)$$

Hence, we calculate the conditional expectation of the $X(t+\tau)$ given the observation set $\theta(t)$. The $X(t+\tau)$ can be expressed as:

$$X(t+\tau) = X(t) + 1_{\{\tau \leq Y_t\}} V(t)\tau + 1_{\{\tau > Y_t\}}(V(t)Y_t + \triangle P(\tau - Y_t)), \quad (2.8)$$

where $Y_t = S_{w(t)+1} - t$ is the time until the node hits the next waypoint, and $\triangle P(\tau - Y_t)$ is the displacement of the node after reaching the next waypoint until time $t+\tau$. Figure 2.1 explains (2.8). Expected value of the $\triangle P(\tau - Y_t)$ is equal to zero (Lemma 2.2 in the Appendix). Thus, we need to calculate only the conditional expectation of the Y_t given the $\theta(t)$. Since we observe the system at a random point in time, the Y_t is the residual time of the renewal process $\{S_w\}_{w\in\mathbb{N}}$. Note that the segment duration on which the node is at a random time t is not distributed according to the distribution of the intervals $S_{w+1} - S_w$. This is because a long segment is more likely to be "intercepted" by our observation than a short one [Kle75]. We use Palm calculus (Chapter 12,[Bou]) to relate these different viewpoints and to calculate the distribution of the residual time in the concrete case. Since the calculation for Theorem 2.3 is trivial, we give further calculations only for Theorems 2.1 and 2.2 in the Appendix.

We were unable to calculate mobility predictors in Theorems 2.1 and 2.2 in a closed form. Instead, in the simulations we use predictors given in Approximations 1 and 2 in the Appendix.

2.2 A LER Algorithm for the Random Waypoint Model

We first describe GREASE, an existing instance of a LER algorithm presented in [GV03]. Then we introduce GREASE-RWP, a new LER algorithm that takes into account the features of the RWP model described in the previous section.

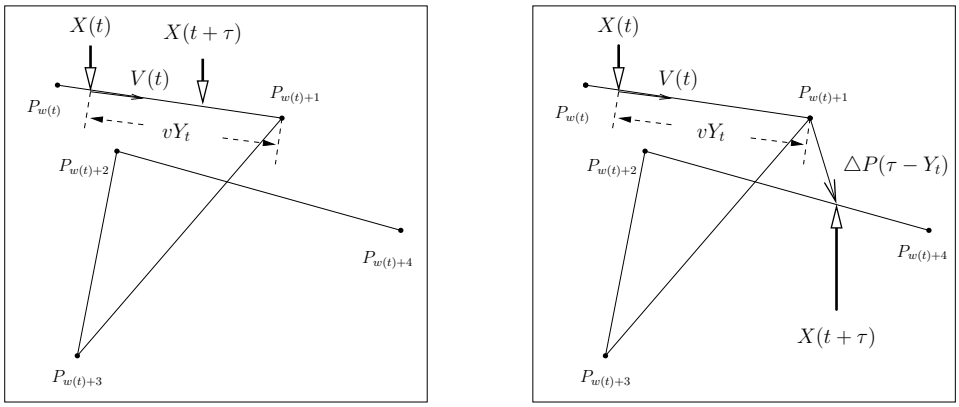


Figure 2.1: Explanation of (2.8): one possible movement of a node and its positions at times t and $t + \tau$ for different values of τ.

2.2.1 GREASE

Algorithm 1 shows GREASE. Notation is the following. Nodes are indexed by $1, 2, ..., n$ where n is the number of nodes. We focus on a single destination node with index 1 and assume w.l.g. that a packet is sent to the destination at time $t = 0$. The $X_i(t)$ is the position of the node i at time t. The $T_i(t)$ is the age of last encounter of the node i and destination.

Initially, the packet is at its source. Then, a search is performed around the source to find a LE entry for the packet's destination that is two times younger than the source's LE entry (i.e., an entry of a node i for which $T_i(0) \leq T_s(0)/2$). The packet is headed towards the entry's LE location $X_1(-T_i(0))$. This location is called an anchor point. When the packet comes to the anchor point it performs another search to find a new anchor point. If the packet encounters a node that has a more recent estimate of the destination's location than the anchor point the packet is currently headed to, then that estimate is assumed to be the new anchor point. The procedure repeats until the packet finds its destination node. Note that tt is not prescribed a particular routing algorithm for the packet to get from one anchor point to the next; any position-based routing algorithm could be used for this purpose (cf. Section 1.3).

Algorithm 1: GREASE

1 Set $T_0 := T_s(0)$, $Y_0 := X_s(0)$, $k := 0$.
2 Repeat
3 Search the nodes around Y_k in order of increasing distance until a node i is found such that $T_i(0) \leq T_k/2$.
4 Let $T_{k+1} = T_i(0)$, and $Y_{k+1} := X_1(-T_{k+1})$ be the new anchor point.
5 While not at Y_{k+1}

6 Route packet: find next hop j towards Y_{k+1} and forward packet to j.
7 If $T_j(0) \leq T_{k+1}$, then $T_{k+1} := T_j(0)$, $Y_{k+1} := X_1(-T_{k+1})$.
8 End while
9 $k++$.
10 Until $Y_k = X_1(0)$.

2.2.2 GREASE-RWP

The notation is the same with the following additions. The $\theta_i(t)$ is the observation set of the node i at time t. The optimal mobility predictor of the destination position at time t is denoted by $\widehat{X_1}(t)$. Recall that T_r is the relaxation time.

Algorithm 2: GREASE-RWP

1 Set $Y_0 := X_s(0)$, $T_0 := E[T_r]$, $k := 0$.
2 Repeat
3 Search the nodes around Y_k in the order of the increasing distance until a node i is found such that $T_i(0) < T_k$.
4 Let $T_{k+1} = T_i(0)$, and $Y_{k+1} := \widehat{X_1}(0) = f(\theta_1(-T_{k+1}))$ be the new anchor point.
5 While not at Y_{k+1}
6 Route packet: find next hop j towards Y_{k+1} and forward packet to j.
7 If $T_j(0) < T_{k+1}$, then $T_{k+1} := T_j(0)$, $Y_{k+1} := \widehat{X_1}(0) = f(\theta_1(-T_{k+1}))$.
8 End while
9 k++.
10 Until $Y_k = X_1(0)$.

The main new features in GREASE-RWP in comparison with GREASE [GV03] are aggressive initial search and prediction.

Aggressive initial search means that a source searches for an encounter with the destination younger than average relaxation time. As we saw in Sect. 2.1.1 average relaxation time is of the order of a few segment times. This means that the packet immediately goes to the few last segments of the destination movement. Thus the packet avoids the useless walking over the network area.

Prediction means that the packet is routed to the predicted destination position rather than to the location of an encounter. It forces the packet to go to the predicted end of the segments. Thus the packet takes a shortcut between the segments. Prediction is incorporated into LER as follows. Every node tracks its own mobility. When two nodes encounter, they exchange some observations about their own mobility (e.g., their current speed and direction) along with ”hello” messages. Each node records these observations in its LE table (cf. Fig. 2.2). They are used to calculate the optimal mobility predictor.

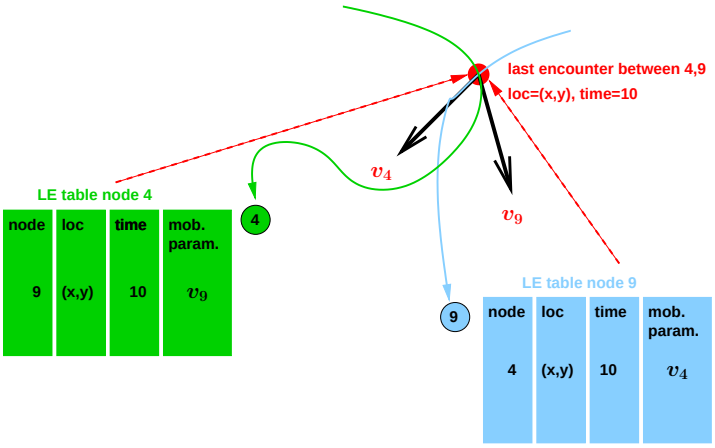


Figure 2.2: A *last encounter table* in every node remembers both the location and time of the last encounter with every other node in the network and some observations about every other node's movement at the time of the last encounter (in this example, speed vector).

2.3 Simulation results

We have performed extensive simulations to evaluate the efficiency and scalability of GREASE-RWP. By the term efficiency, we mean how much in average the routing cost is larger then the shortest route. By the term scalability, we mean how the efficiency scales with increasing the network size.

Nodes move on a torus of surface n according to the RWP model. Initially the nodes are placed uniformly in the torus. We let them to move for a sufficient warm-up period so that a fair proportion of node pairs have encountered at least once. Then, we assume that the nodes are frozen for the time of the routing of the packet.

Routing of the packet is performed through the GREASE (Algorithm 1), GREASE-RWP (Algorithm 2 in this paper), GREASE-M and GREASE-A algorithms. GREASE-M is GREASE with mobility prediction. We obtain it by changing the step 4 in GREASE with the step 4 in GREASE-RWP. GREASE-A is GREASE with aggressive initial search. We obtain it by changing the step 1 in GREASE with the step 1 in GREASE-RWP. The Mi (i=1,2,3) denotes respectively the predictors from Theorems 2.1-2.3. If nothing is specified the predictor M2 is used.

At every time t, we assume that connectivity is given by the Delaunay graph generated by the set of points $\{X_i(t)\}$. This is equivalent to generating the Voronoi tessellation of the set of points $\{X_i(t)\}$, such that every node $X_i(t)$ is the center of a Voronoi cell, and is connected to the center nodes of its adjacent cells. Each node updates the entries in its LE table for its directly connected neighbors.

The advantage of this topology over other topologies (e.g., k nearest neighbors) is that we are guaranteed that a node always has a neighbor that is closer to the

destination (except when that destination is already in the first node's Voronoi cell). Therefore, a packet can always make progress towards its anchor point, and we do not have to deal with backtracking, avoiding routing loops, etc. This allows us to focus on the main issue at hand, i.e., the quality of computed routes based on diffused information about last encounters.

The main metric we evaluate is the relative cost of the routes compared with the cost of the shortest path route. The cost of a route is the total number of transmissions (or hops) necessary to transmit a packet from a source to a destination. It includes both transmissions of the actual packet from a sender to a receiver node to make progress towards its destination, as well as transmissions necessary for a "search" packet to collect information from surrounding nodes to make the next routing decision. This metric therefore captures the relative penalty incurred for not having the exact position of the destination available.

Figure 2.3 shows an example of a route computed by GREASE and GREASE-RWP for the same source-destination pair. We see that GREASE-RWP achieves the shorter route than GREASE by increasing the cost of the initial search and by taking shortcuts with the help of mobility prediction.

In Fig. 2.4(a) and 2.4(b) we give the relative cost conditional on the distance between the source and destination. This provides an indication whether the relative quality of the routes increases or decreases as the routes get longer.

Figure 2.4(a) shows the benefit of mobility prediction if different observation sets are used. If more information about node mobility is available, improvement of GREASE is better.

Figure 2.4(b) shows that the penalty of the GREASE-RWP algorithm because of the uncertainty of the destination location is only 2.5 times greater than the shortest route (i.e., the ideal case where the destination location is known). This is more than 35 % better than GREASE. Also, we see that aggressive initial search or mobility prediction alone significantly increase the efficiency.

In Fig. 2.5 we give dependence of the average relative cost of routes on the number of nodes n. This provides an indication whether the relative cost of the routes is scalable with the network size.

2.4 Conclusion

In this chapter, we have shown that efficient and scalable last encounter routing under random waypoint mobility is possible. We have achieved this by devising a new instance of the GREASE algorithm, which differs from the version reported in [GV03] in two respects. First, we have exploited the inherent predictability of nodal movement in the RWP model over short time-scales. When a packet looking for its destination picks up a more recent encounter, it can compute a predicted location for the destination that is better on average than the location of that encounter itself. Second, we account for the fact that the RWP model has a very short relaxation time, by forcing a low target age for the initial search. Thus the packet

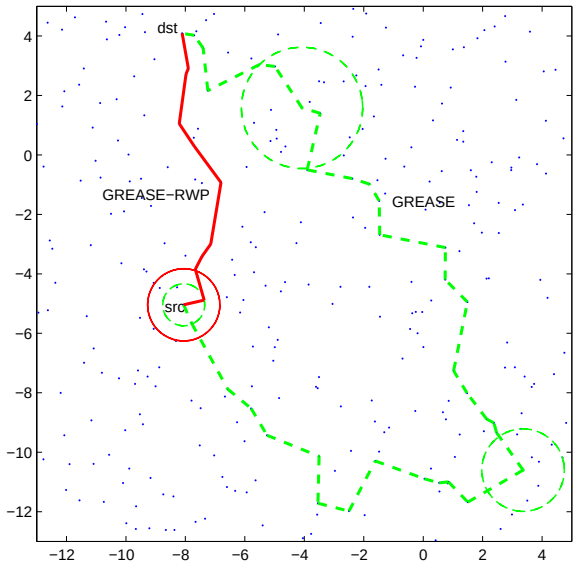
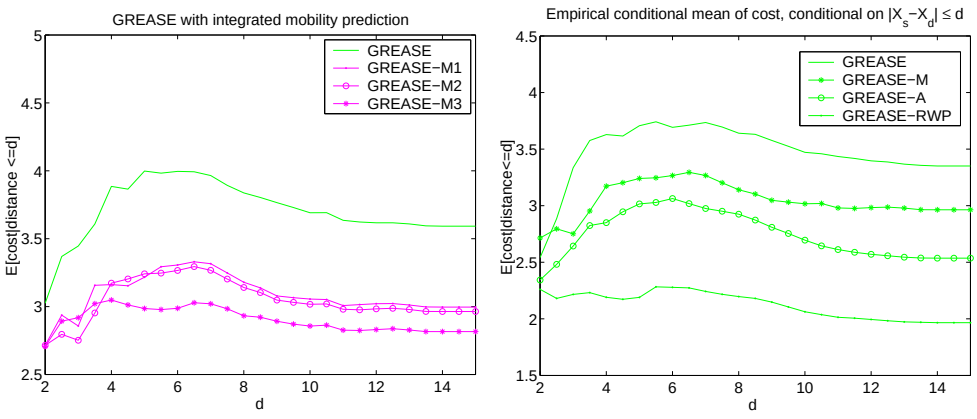


Figure 2.3: A sample route computed by GREASE and GREASE-RWP for the same source-destination pair.



(a) (b)

Figure 2.4: a) Impact of the different observation sets in mobility prediction on the efficiency of GREASE. b) Efficiency of GREASE, GREASE-M (mobility prediction), GREASE-A (aggressive initial search) and GREASE-RWP (mobility prediction and aggressive initial search).

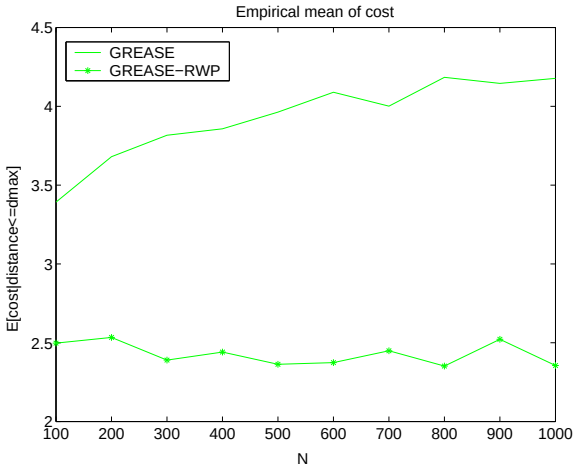


Figure 2.5: Scalability of the GREASE-RWP algorithm.

avoids using outdated past encounters that are independent of the destination's actual position.

We have performed simulations that show the following results. First, the GREASE-RWP algorithm achieves drastically better performance than the "non-RWP" version reported in [GV03]. Specifically, the total average route cost are slightly more than twice as long as the shortest path. This is quite remarkable, given that no resources were invested to track the rapid change in the network topology due to RWP mobility. Second, our results show, as we would expect, that the benefit of prediction depends on what observations are available about a node's mobility. It should be pointed out that if we extended prediction beyond the next waypoint, the performance would be further improved. Third, the cost of GREASE-RWP routes relative to the shortest path does not seem to increase as we scale up the network size n. Therefore, we believe that a similar scaling result as shown in [GV03] for the random walk holds for the RWP model as well.

Predictability is a mobility feature recognized in many realistic scenarios. For example, the limitation of the speed of nodes makes a mobility process predictable over at least a short time scale. Another example is that nodes may have some locations that they visit more often than others, as mentioned in Section 1.2, which is exploited by many routing mechanisms designed for the partitioned networks, as we see in Section 1.3. The novelty of our work is that we consider the prediction in the RWP model, and how to incorporate this prediction into LER, both of which have not been previously considered.

The prediction in the RWP model exploits a specific nature of the model, where a node moves with the same speed between two waypoints, which allows us to precisely predict where the node can be for a short time scale. But, the fact that a next waypoint does not depend on the previous waypoints makes it impossible to predict future node's movements on a longer time-scale, beyond the next waypoint.

We argue that these properties are indeed present to some extent in some realistic scenarios of interests, but that they are too specific. Therefore, as a next step in our thesis, we look at realistic mobility processes in order to find common properties in many realistic applications that we can exploit in routing.

Appendix

Auxiliary Results

Lemma 2.1 *The sequence of the segment lengths L_ws has the following pdf:*

$$f_L(l) = \begin{cases} \frac{2l\pi}{a^2} & \text{if } 0 \leq l \leq \frac{a}{2} \\ \frac{2l\pi}{a^2} - \frac{8l}{a^2} \arccos \frac{a}{2l} & \text{if } \frac{a}{2} \leq l \leq \frac{a\sqrt{2}}{2} \\ 0 & \text{otherwise} \end{cases} . \tag{2.9}$$

Lemma 2.2 *The expected displacement of a node from a waypoint is equal to zero, i.e., the expectation of the random variable $X(S_w + \triangle t) - X(S_w)$ is equal to zero, for every w and $\triangle t > 0$.*

Optimal Predictors

Theorems 2.1 and 2.2

Proof:

Using (2.7) and (2.8), the optimal mobility predictor is:

$$\begin{aligned} \hat{x}(t+\tau) &= E[X(t+\tau) \mid \theta(t)] = \\ &= x(t) + v(t)\tau P(Y_t \geq \tau \mid \theta(t)) + v(t)E[Y_t 1_{\{Y_t < \tau\}} \mid \theta(t)] + \\ &\quad + E[\triangle P(\tau - Y_t) 1_{\{Y_t < \tau\}} \mid \theta(t)]. \end{aligned}$$

The last term of this equation is equal to 0 (Lemma 2.2). Thus, we obtain:

$$\hat{x}(t+\tau) = x(t) + v(t)\tau P(Z \geq v\tau) + \frac{v(t)}{v} E[Z 1_{\{Z > v\tau\}}],$$

where Z is the random variable with the same distribution as vY_t given $\theta(t)$. Next, we calculate the distribution of Z separately for Theorems 2.1 and 2.2.

In Theorem 2.1, the set of observations is $\theta(t) = \{X(t) = x(t), V(t) = v(t)\}$. The residual time Y_t depends only on the renewal process S_w. The S_ws depend only on the segment lengths L_ws because of the constant speed. The L_ws do not depend on $X(t)$ since every point in the torus is the same. Therefore, the Y_t does not depend on $X(t)$. As mentioned previous, we neglect that the Y_t depends on the direction of the speed vector, thus the Y_t does not depend on $V(t)$. Hence, the Z has the same distribution as vY_t. The pdf of residual time Y_t is equal to [Kle75]:

$$f_Y(y) = \frac{1 - F_R(y)}{m_R},$$

where $F_R(x)$ is the cdf of the intervals $R_w = S_{w+1} - S_w$ and m_R is the mean of R_w. Since $R_w = L_w/v$, the pdf of Z is:

$$f_Z(z) = \frac{1 - F_L(z)}{m_L},$$

where $F_L(z)$ is the cdf of L_ws (Lemma 2.1), and $m_L = a/6(\sqrt{2} + \ln(1+\sqrt{2}))$ is its expected value. We obtain the pdf of Z given by (2.3).

In Theorem 2.2, the set of observations is $\theta(t) = \{X(t) = x(t), V(t) = v(t), P_{w(t)} = p\}$. As in the previous case, the residual time Y_t does not depend on $X(t)$ and $V(t)$. Knowing both $X(t) = x(t)$ and $P_{w(t)} = p$ we know that the previous waypoint was at time $t - d(p, x(t))/v = t - t_0$. This means that there is a point at time $t - t_0$ and that there is no point in interval $(t - t_0, t)$. This second condition is equivalent to $Y_{t-t_0} \geq t_0$. If we denote the conditional probability given that there exists a point at time t as P^t then the cdf of Z is equal to:

$$\begin{aligned} P(Z \leq z) &= P(vY_t \leq z \mid X(t) = x(t), V(t) = v(t), P_{w(t)} = p) \\ &= P(vY_t \leq z \mid \text{there exists a point at time } t - t_0, Y_{t-t_0} \geq t_0) \\ &= P^{t-t_0}(vY_t \leq z \mid Y_{t-t_0} \geq t_0) = P^0(Y_0 \leq \frac{z}{v} + t_0 \mid Y_0 \geq t_0) \\ &= P^0(S_1 \leq \frac{z}{v} + t_0 \mid S_1 \geq t_0). \end{aligned}$$

Using the result of Palm calculus that $P^0(S_1 \leq x) = P^0(R_1 \leq x) = F_R(x)$ ([Bou]), we obtain:

$$P(Z \leq z) = \frac{F_R(\frac{z}{v} + t_0) - F_R(t_0)}{1 - F_R(t_0)}.$$

Since $R_w = L_w/v$, the cdf of Z can be expressed as:

$$P(Z \leq z) = \frac{F_L(z + d_0) - F_L(d_0)}{1 - F_L(d_0)},$$

and we obtain the pdf of Z given by (2.5). □ □

Approximation 1

We are unable to compute in a closed form the predictor in Theorem 2.1. Therefore, we use the following mobility predictor:

$$\begin{aligned} \widehat{x}(t+\tau) &= \begin{cases} x(t) + \frac{v(t)}{v}(v\tau - \frac{c(v\tau)^2}{2} + \frac{c\pi(v\tau)^4}{12a^2}) & \text{if } 0 \leq v\tau \leq \frac{a}{2} \\ x(t) + \frac{v(t)}{v}E[Z] & \text{if } v\tau > \frac{a\sqrt{2})}{2} \end{cases} \\ \widehat{x}(t+\tau) &\approx x(t) + \frac{v(t)}{v}E[Z] \quad \text{if } \frac{a}{2} \leq v\tau \leq \frac{a\sqrt{2}}{2}. \end{aligned}$$

The mean value of the Z is given by:

$$E[Z] = \frac{E[L]}{2} + \frac{Var(L)}{2E[L]},$$

where the mean $E[L]$ and the variance $Var(L)$ of the L_is are equal to $E[L] = a/6(\sqrt{2} + \ln(1+\sqrt{2}))$ and $Var(L) = \frac{a^2}{6} - E[L]^2$, respectively. We make an approximation in the region $\frac{a}{2} < v\tau < \frac{a\sqrt{2}}{2}$ by neglecting the probability that there is not any waypoint in the period of $v\tau$.

Approximation 2.

We are unable to compute in a closed form the predictor in Theorem 2.2. Therefore, we use the following mobility predictor:

$$\widehat{x}(t+\tau) = x(t) + \frac{v(t)}{v}\frac{\pi l_0^3 - \pi(v\tau+l_0)^3 + 3v\tau a^2}{3(a^2 - l_0^2\pi)} \quad \text{if } (l_0, v\tau+l_0) \in [0, \tfrac{a}{2}]^2$$

$$\widehat{x}(t+\tau) \approx \begin{cases} x(t) + v\tau & \text{if } 0 \le l_0 \le \frac{a}{2},\ \frac{a}{2} - l_0 \le v\tau \le E[Z] \\ x(t) + E[Z] & \text{if } 0 \le l_0 \le \frac{a}{2},\ \frac{a}{2} - l_0 \le v\tau,\ v\tau > E[Z] \\ x(t) & \text{if } l_0 > \frac{a\sqrt{(2)}}{2} \end{cases}.$$

The mean value of the Z is given by:

$$E[Z] = \frac{3a^2(E[L] - l_0) + \pi l_0^3}{3(a^2 - l_0^2)},$$

where the mean value of the L_is is equal to $E[L] = \frac{a}{6}(\sqrt{2} + \ln(1+\sqrt{2}))$. We make an approximation in the region $0 \le l_0 \le \frac{a}{2}$, $\frac{a}{2} - l_0 \le v\tau \le E[Z]$ by neglecting probability that there is a waypoint in the period of $v\tau$. We make also an approximation in the region $0 \le l_0 \le \frac{a}{2}$, $\frac{a}{2} - l_0 \le v\tau$, $v\tau > E[Z]$ by neglecting probability that there is not any waypoint in the period of $v\tau$. In the region $l_0 > a/2$ we do not make mobility prediction because we are unable to compute $E[Z]$. The probability of the appearance of this last case during the operation of the LER algorithm is small because the probability that $l_0 > a/2$ is small. Thus, this will have a small effect on the predictor.

Chapter 3

Modeling Stable Clusters in Mobility

In this chapter, we look at realistic mobility processes and ask the following questions. What properties do they possess? Which ones are common in many realistic applications? Which ones can we exploit in routing?

We observe three large-scale realistic mobility traces and we identify the following properties:

- a spatial distribution of the nodes is heterogenous rather than homogenous, i.e., there are both dense and sparse regions of connectivity;
- a network is often partitioned, which is a consequence of this heterogenous spatial distribution;
- there exist islands (clusters) of dense connectivity, which we call concentration points (CPs);
- the CPs, and the average flows of nodes between CPs, typically remain stable over relatively long time-scales.

These observations led us to devise two synthetic mobility models that posses the above mentioned properties. We call these models: 1) a graph-based (G) model and 2) a heterogenous random walk (H) model.

We design the G-model with the main purpose of using it for fitting data to a model, i.e., for inferring a stable topology of CPs from mobility data by COGRAD. Hence, the G-model is a "rich" model in a sense that it has many parameters. Moreover, in order to capture the essence of the problem, it makes simplified assumptions about the nodes' connectivity. The G-model assumes a "sharp" connectivity inside and outside CPs, i.e., the nodes inside a CP are all connected, and the nodes outside of CPs are not connected to any other node.

We design the H-model for the purpose of modeling the mobility of the nodes in order to evaluate both COGRAD and IH. Our design goal is to have a parsimonious

model with as few parameters as possible and to have a model with more realistic connectivity assumptions than the G-model. That is to say, we want to allow that not all nodes inside a CP are necessarily connected and that nodes outside CPs may be connected to other nodes as well.

In our design and evaluation of IH and COGRAD, we first start modeling the mobility of nodes by the G-model in order to concentrate on the essence of the problem by abstracting away the complexity of real mobility. Then, we consider IH and COGRAD under the relaxed connectivity assumptions in the H-model. Note that in the both cases COGRAD uses the G-model to infer the CP topology.

This chapter is organized as follows. In Section 3.1 we describe realistic mobility data sets and our findings of their common properties. In Section 3.2 we define the G-model and we derive stable CPs from the data sets. In Section 3.3 we define the H-model. We also calculate the spatial stationary distribution of nodes in the H-model and we discuss how to perform simulations.

3.1 Properties of Real Mobility [1]

For the purpose of our study we use three large GPS-based mobility data sets.

The two data sets are mobility traces of taxi cabs from two cities: Warsaw, Poland and San Francisco, USA. The Warsaw data set contains GPS coordinates of 825 taxis collected over 92 days in the Warsaw agglomeration area (25x40 km). The San Francisco data set contains GPS coordinates of approximately 500 taxis collected over 30 days in the Bay area (14x25 km). In both cases each taxi is equipped with a GPS receiver and sends a *location update* (timestamp, identifier, geographical coordinates) to a central server. Updates are not periodic, rather they are irregular. In the case of the Warsaw data set the updates are infrequent - they can be as frequent as a few per hour or only a few per day. In the case of the San Francisco data set the locatio-updates are quite frequent - the average time interval between two consecutive location updates is less than 10 sec, allowing us to accurately interpolate node positions between location-updates.

The third data set is less detailed than the taxi data sets. It contains GPS traces collected by mobile-phone users subscribed to the Nokia Sports-tracker service[2]. Mobile-phone users can upload workout and activity traces to the Nokia Sportstracker web site to store and share with others. So, the data set provides only occasional snapshots of a person's long-term mobility, i.e., it only consists of traces obtained during some activities (running, walking, cycling, etc.). This data set is

[1] The results presented in this section as well as in Sections 3.2.1 and 3.3.2 are contributions of Michal Piorkowski [Pio09]. These results were obtained during our joint work for the purposes of i) observing important properties of realistic mobility, and ii) validating our modeling of the observed mobility properties. More precisely, finding realistic mobility data sets, the algorithms for the performed data analysis as well as the data analysis are all contributions of Michal Piorkowski. However, mobility modeling is my contribution. We include the results contributed by Piorkowski in order to help a reader in understanding better the presented material.

[2] `http://sportstracker.nokia.com/`

larger than the taxi data sets in terms of the number of mobile nodes, and it also covers a much larger geographical region and time period (more than one year). For the purpose of this study we focused on subsets from Helsinki, Finland (3757 distinct GPS traces, 11x8 km), Stockholm, Sweden (1056 distinct GPS traces) and London, UK (2488 distinct GPS traces).

Next, in these data sets we observe that the node distribution is spatially heterogenous and temporally stable. I.e., we observe that there exist regions where the node density is much larger than on average which are persistent over different days. We also observe that because of this spatially heterogenous node distribution, the network becomes partitioned where many partitions contain relatively large number of nodes.

3.1.1 Spatial Distribution

To check for the existence of stable CPs we apply the following heuristic. First, we superimpose a grid of equal-sized cells on the area of the Warsaw and San Francisco agglomerations. Then, for each day d and each cell (k, l) we find the *normalized population* - $f(k, l; d)$, interpreted as the empirical probability that a random update falls into the cell (k, l) on day d. Our analysis shows the following:

The Spatial Distribution is Heavy Tailed

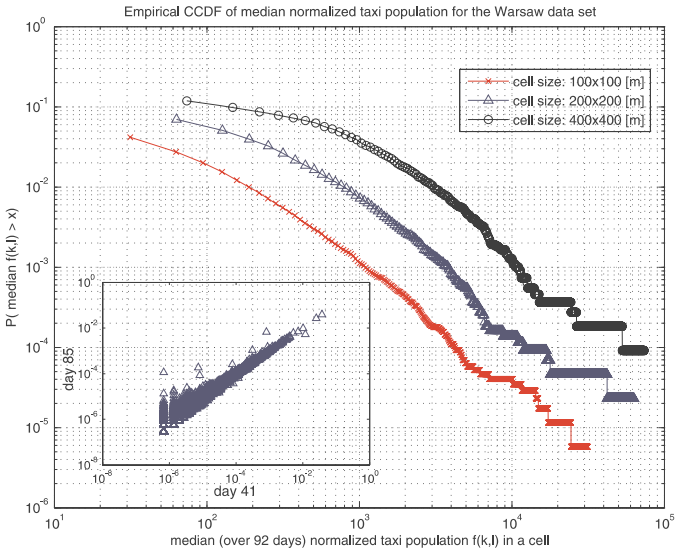
Figures 3.1(a) and 3.1(b) show the empirical complementary cumulative distribution function (CCDF) of $f(k, l; d)$ for the two data sets - from Warsaw and from San Francisco respectively. Both distributions have heavy tails, which implies that some cells in both cities have a population density much above the average.

The Spatial Distribution is Stable Over Time

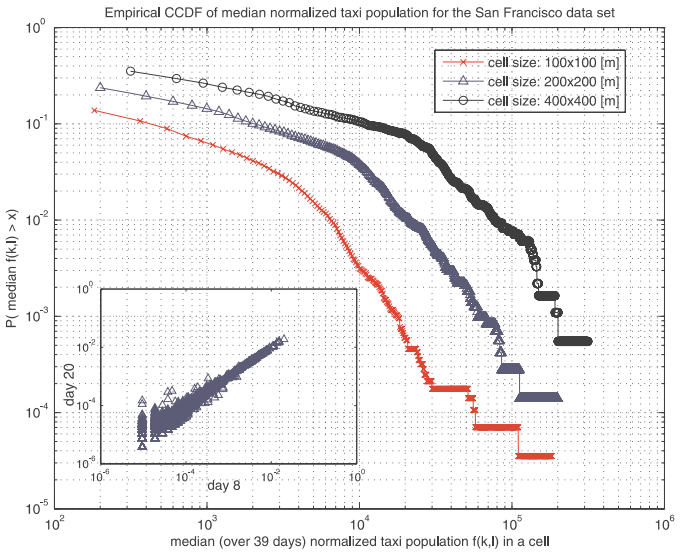
Figures 3.1(a) and 3.1(b) insets show scatter plots of $f(k, l; d)$ for one randomly chosen pair of days (d_1, d_2). In both cases we observe significant clustering along the diagonal, which means that the spatial distribution on different days tends to be strongly correlated. Furthermore, we observe that the more densely populated cells (upper-right quadrant) tend to be particularly close to the diagonal, which is a good visual confirmation of our hypothesis. We observe the same behavior for other pairs of days.

3.1.2 Partitioned Connectivity

Let us first define the *connectivity graph*. Here again we assume a (scaled) unit disk model for connectivity. The mobile nodes and the corresponding wireless links define the *connectivity graph* $G(V, E)$, where $V(G)$ is the set of mobile nodes and $E(G)$ is the set of radio links between mobile nodes, i.e., $E(G) = \{e = (i, j) | d_{ij} \leq r\}$. We define H_k as a strongly connected component of G, with $C(G) = \{H_1, H_2 \cdots, H_K\}$ the set of all components, i.e., $G = \bigcup_{k=0}^{k=K} H_k$.



(a) Warsaw data set



(b) San Francisco data set

Figure 3.1: Empirical CCDF of $f(k,l;d)$ for the entire period for three levels of discretization for two data sets (a) Warsaw and (b) San Francisco. Insets in (a) and in (b) shows the scatter plot of $f(k,l;d)$ on two random days - each point on the plot corresponds to a density in a cell (k,l) for different days.

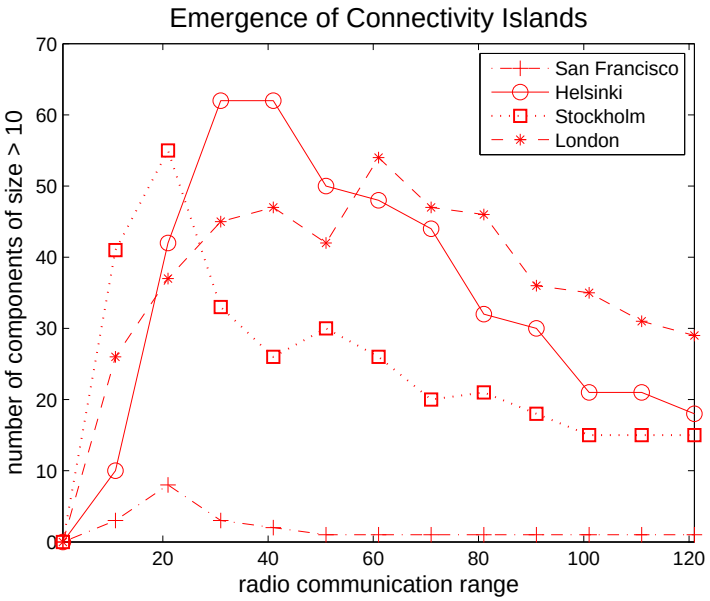


Figure 3.2: Emergence of connectivity islands in the connectivity graph - each data point represents the number of components of size larger than 10 in the connectivity graph $G(V, E)$ for different values of r (scale unit is meter $[m]$).

To show that islands of connectivity may emerge we study how the structure of the connectivity graph changes with increasing r. More precisely, we focus on the size of components $|H_k|$ present in the connectivity graph $C(G)$ generated at a random time instant from both the taxi cab and Nokia Sportstracker data sets.

For this experiment, we *densify* our trace in the following way. We assume that the sampled nodes are a representative subset of the overall traffic, and that their movements are stationary and ergodic. Under this assumption, we can generate a denser sample of instantaneous node locations (in our case, 5000 nodes) by sampling uniformly at random from the entire data set. This denser sample brings out more detail in the connectivity graph.

In Figure 3.2 we show the results for four cities: San Francisco, Helsinki, Stockholm and London. We observe a clear trend in all the cases. The number of components rises quickly with the communication range r towards a maximum; past this critical value, the number of components starts to decrease, because smaller components start coalescing into larger ones. However, this decrease tends to be slow; even when r becomes a multiple of the critical value, we still have many components left. This is because node locations are distributed non-uniformly in space, which prevents percolation into a single giant cluster when r grows. Thus, we conclude that disconnected network topologies with a large number of components seem to be a robust phenomenon that persists over a wide range of radio ranges.

3.2 Graph Based Model (G-model)

Given the observations about the presence of stable CPs in realistic data, we now define an idealized mobility model that embodies CPs. This model is to be used for fitting data to a model, thus it is rich with parameters and simplistic regarding assumptions about a detailed connectivity of nodes.

The network topology is given by a directed connected graph $G(V, E)$ whose vertex set V represents the CPs, and whose edges E describe the possible movements of nodes between CPs. We call this graph *a CP graph*. There are n nodes that move on this graph. At every time t, every node i is either located at one CP, or is en route between two CPs. We denote the current position of node i by $X_i(t) \subset V \cup \phi$, where ϕ denotes that it is en route between two CPs. We assume that nodes located at the same CP can communicate with each other (either directly or through multi-hop), whereas nodes at different CPs cannot. We call $B_i(t)$ the set of neighbors of node i at time t, i.e., $B_i(t)$ is the set of nodes located at the same CP as i (including i), and if node i is en route between two CPs then $B_i(t) = \{i\}$ (cf. Figure 3.3).

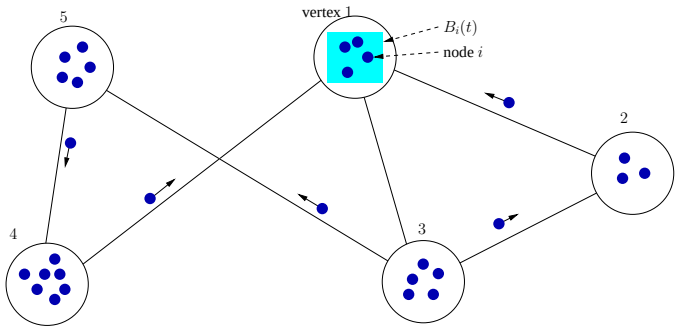


Figure 3.3: Nodes move on a graph $G(V, E)$, which describes the network topology in terms of its CPs and the ways nodes can move between them.

3.2.1 Inferring the CP Graph from a Mobility Trace

Our goal is to infer the CP graph from real data in order to use the G-model with this inferred CP graph for an evaluation of our routing algorithm (Section 4.5). We use different procedures in the Warsaw and San Francisco data sets. This is because the San Francisco data set consists of the frequent location updates, which is not the case for the Warsaw data set. Thus, the San Francisco data set allows us to make a realistic analysis of dynamics of the nodes' connectivity, i.e., to analyze how the nodes' connectivity evolves over time and then, based on this, to infer CPs. This is impossible to do in the Warsaw data set, so we derive an approximate heuristic for this. Note that in our further investigations we use only the CP graph inferred from the Warsaw data set, because there are only a small number of CPs in the San Francisco data set.

The Warsaw Data Set

In order to find CPs, we first define a *cluster* at day d as a cell for which $f(x, y; d) > 5\%$. The reason we choose 5% as the threshold is to ensure that in every CP there are at least 15 vehicles (see 3.1.1). Using such a small threshold allows us to identify clusters that would be more visible for regular and frequent location updates. We identify 174 clusters. Here we define a CP as a cluster that is present for more than 20 days (see 3.1.1). In result we find 79 CPs within the whole city (cf. Figure 3.4).

Inferring stable flows between CPs from a such data set is not trivial because of very irregular and infrequent location updates. If the period between two successive location updates of a taxi is too long, the taxi might visit several CPs during this time. Thus, we may miss direct flows of nodes between CPs and, because of this, we may consider falsely a non-direct flow as direct. For example, a graph G' where the edges are all observed flows between CPs over all taxis and over the entire data set is an almost complete graph. In order to prune the false flows, we propose the following heuristic. For each edge $(u, v) \in G'$ we find a minimum travel time $\tau_{min}(u, v)$ over all taxis and over the entire data set. Then we delete an edge (u, v) from G' if there exists a path between u and v, $p(u, v) = (u, w_1, w_2, ..., v)$, such that $\tau_{min}(u, v) > \sum_{e \in p(u,v)} \tau_{min}(e)$. Instead of an almost complete graph, we obtain a graph G for which the average vertex degree is 3.2.

The inferred CP graph G is shown in Figure 3.4. The resulting topology resembles a spider net, which is consistent with the topology of the city of Warsaw, where most of the important institutions and centers of activity are located downtown.

The San Francisco Data Set

In the San Francisco data set the location updates are frequent enough that it is possible to study how the nodes' connectivity graph H_t changes in time. Studying H_t allows us to identify CPs, i.e., regions where node density is higher than average and is stable over time. In order to find such regions, we could apply one of the well-known data clustering algorithms, e.g. k-means clustering [The03] - for every time instant one can find such regions and then identify which of them last for a long time. This approach is used in the works of [KWSB04] and [AS03] to extract significant users' locations where the users tend to spend much more time on average than in other places. But, we take another approach - we identify connected components with a relative large number of nodes that last for a long time. Here we rely on an intuition that in highly populated regions nodes should form stable connectivity islands. The main advantage of our approach over the clustering algorithms is that our approach does not require nodes' positions. Note also that the works of [KWSB04] and [AS03] solve a slightly different problem, they search for significant locations of an individual user, and we search for collective significant locations that many users visit at the same time.

As nodes are mobile and the connectivity graph changes dynamically over

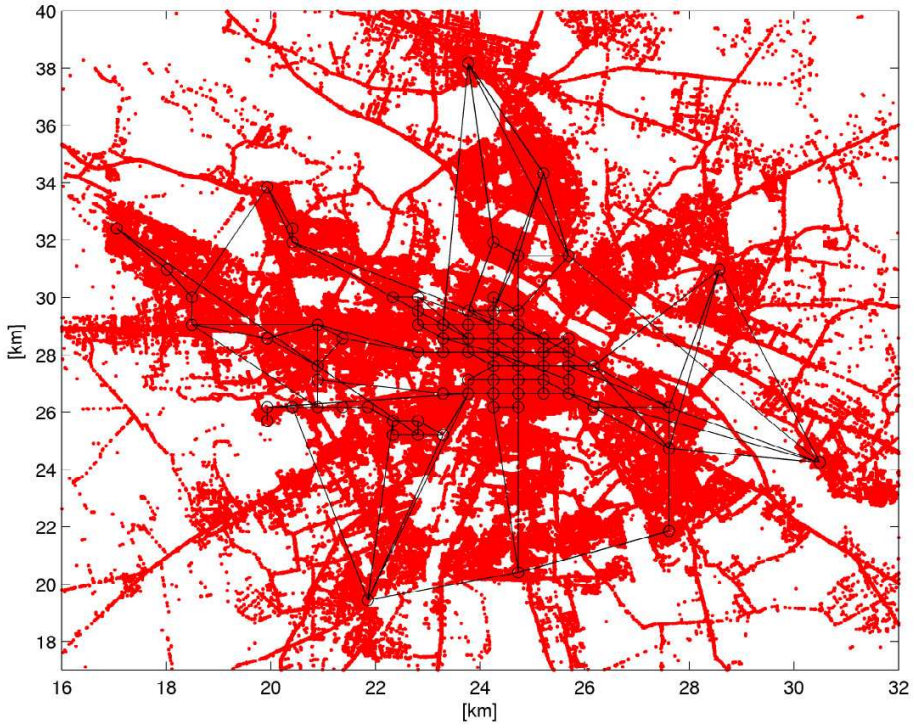


Figure 3.4: The red dots represent superimposed location updates of 825 taxis over 92 days taken from the data set. The black circles represent 79 CPs and black lines show taxi flows between these CPs. Both CPs and flows are extracted from the data set.

time, the key question is how to establish a correspondence between components at subsequent time steps. For this, we introduce the concept of *cluster labeling*. This means assigning an unique identifier y (we call it a label) to every connected component, in such a way that the two components at the consecutive time moments with the same label correspond to each other. To do this we devise a Centralized Cluster Labeling (CCL) algorithm. Thus, sequences of corresponding components over time are uniquely identified by a label.

We expect CPs to be stable over time and have a large size, thus we seek labels that last for a long time and are owned by a large (on average) number of nodes. Note that the CCL algorithm does not use nodes' positions - it uses only the connectivity graph, which is one advantage over the data clustering algorithms, as mentioned above. Because of this, there is no information whatsoever about the location of the CPs. However, given the evidence presented below, we believe that stable clusters should appear at certain fixed locations.

In order to extract CPs from collection of connectivity graphs, by applying the CCL algorithm, we use the part of the San Francisco traces - approximately 500 taxis over 24 hour period. The connectivity graphs are generated every $\Delta t = 10$ seconds for connectivity range $r_c = 300$ meters. We determine which labels can correspond to CPs by using two threshold values, one for the lifetime and second for the size of the label: 1800 seconds and 10 nodes respectively. We found 19 such labels, of which several of them lived even more than 2 hours. However, these 19 labels do not specify 19 distinct CPs. This is because the same connectivity island may re-appear at different time of the day, which cannot be captured by the CCL algorithm.

Thus, we check if a labeled cluster appears in the same area during its lifetime. We superimpose the locations of nodes that own the same label for three different snapshots. We visualize this on Figure 3.5 where the taxis that own the same label (marked with the same type of a marker) cover the same area at different time instants (different colors of the same marker). This confirms our intuition that islands of connectivity are stable in space as well. We also study the evolution of a cluster location. More specifically we look at the diffusion of the large size clusters' center of mass. For every time instance, we compute the center of mass of each large cluster (larger than 20 vehicles). We make a visual test (cf. Figure 3.6) to see how far the center of a cluster moves in time. As can be observed, during a cluster's lifetime its center of mass does not diffuse far from its initial location. We identify four locations of CPs - namely the aquatic park/shopping center, the downtown area, the taxi company premises and the airport. These four locations can be observed also in Figure 3.5.

As we identify only a small number of CPs in the San Francisco data set we do not use this data set in our evaluation of IH and COGRAD, and therefore we do not make an additional effort to identify flows of nodes in this data set. Nevertheless, the results of our analysis justify the presence and relative stability of CPs in the real world. However, keep in mind that the mobility pattern of taxis is very specific and may not give sufficient evidence of other CPs located for example nearby sport

centers, gas stations, movie theaters etc. Thus we believe that given a large set of mobile traces for the same area, which contains GPS location updates of different types of vehicles, it should provide us with more CPs. We also believe that CPs might be more easily observed at a larger scale, e.g. not at the San Francisco agglomeration scale only, but at the whole Bay Area scale. Note also that this is the first attempt to identify CPs where a connected component is chosen to give the evidence for the CP existence. The presented results show that this choice might be too extreme for determining if a CP exists or not.

Figure 3.5: Four CPs in San Francisco identified by the CCL algorithm. Markers represent taxis that belong to CPs. Different types of markers correspond to different CPs: *triangles* - the aquatic park/shopping center, *balls* - the downtown area, *crosses* - the taxi company premises, *squares* - the airport. The color of each marker represents members of a corresponding CP at different time moments.
.

3.3 The Heterogeneous Random Walk (H) Model

Next we define the H-model. The H-model is for modeling the nodes' mobility with stable CPs. Hence, the design goals are: a parsimonious model with small number of parameters, and more realistic connectivity assumptions than in the G-model.

Here, we also calculate the stationary node distribution in the H-model and conditions for the emergence of CPs, and we discuss how to perform a simulation that starts directly from the stationary distribution (i.e., "perfect simulation" [Bou].) We first define our model as a diffusion process, then show how to correct for boundary effects in a discrete-time approximation. In addition, we validate the

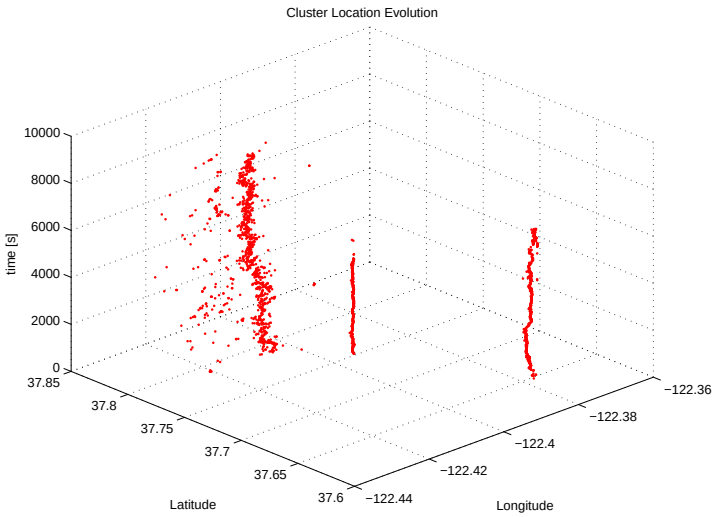


Figure 3.6: Tracking the location of large clusters.

model by comparing its qualitative behavior with a realistic mobility scenario (the San Francisco mobility trace).

3.3.1 Definition

There are n nodes moving independently of each other in a torus of unit side length. We divide the torus into two regions A_l and A_h, where each node performs a two-dimensional Brownian motion with heterogeneous speeds (variances) in these two regions, with a slower speed in A_l than in A_h. The region A_l can be generated, for example, as the union of m random disks of radius r_l, and A_h its complement (cf. Figure 3.7). The difference in variance in the two regions gives rise to different node densities.

More formally ([KT81], Chapter 15), each node moves according to a process $Z = \{Z(t), t \geq 0\}$ taking values in $[0,1]^2$ with the coordinate representation $Z(t) = (Z_x(t), Z_y(t))$, where :

- $Z(0)$ is an initial position in the torus;
- Z_x and Z_y satisfies the following conditions:

$$a_x(z,t) = \lim_{h \to 0} \frac{1}{h} E[Z_x(t+h) - Z_x(t) \mid Z(t) = z] = 0 \tag{3.1}$$

$$b_x(z,t) = \lim_{h \to 0} \frac{1}{h} E[(Z_x(t+h) - Z_x(t))^2 \mid Z(t) = z]$$

$$= b(z) := \begin{cases} \sigma_l^2 & \text{if } z \in A_l \\ \sigma_h^2 & \text{if } z \in A_h \end{cases}, \sigma_l < \sigma_h, \tag{3.2}$$

and similarly for $a_y(z,t)$ and $b_y(z,t)$;

$$a_{xy}(z,t) = \lim_{h\to 0} \frac{1}{h} E[(Z_x(t+h) - Z_x(t)) \times \\ \times (Z_y(t+h) - Z_y(t)) \mid Z(t) = z] = 0. \quad (3.3)$$

Thus, *infinitesimal means* $a_x(z,t)$ and $a_y(z,t)$ of the changes in Z_x and Z_y, respectively, are equal to 0 (3.1). *Infinitesimal variances* $b_x(z,t)$ and $b_y(z,t)$ of the changes in Z_x and Z_y, respectively, are equal to σ_l^2 if a node is in A_l or σ_h^2 if a node is in A_h (3.2), where $\sigma_h^2 = \gamma\sigma_l^2$ with $\gamma > 1$. Note that the variance reflects the speed of the node. Moreover, we constrain ourselves to a process where the infinitesimal changes in Z_x and Z_y are uncorrelated (3.3).

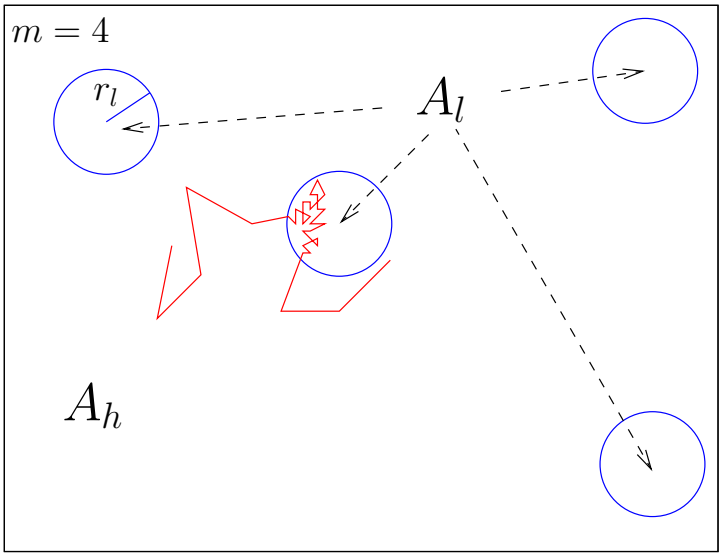


Figure 3.7: Heterogeneous Random Walk - nodes move more slowly in region A_l than in A_h.

The following table summarizes the parameters of the model.

n	# of nodes
m	# of disks (clusters)
r_l	disk radius
$\gamma = \sigma_h^2/\sigma_l^2$	ratio between variances in A_l and A_h
r	connectivity range

Stationary Distribution and Connectivity

Lemma 3.1 *A node's position in the stationary regime has pdf*

$$f(z) = \beta/b(z),\ \beta = \frac{1}{|A_l|/\sigma_l^2 + |A_h|/\sigma_h^2}, \quad (3.4)$$

where $|A|$ is the surface area of the region A.

Proof: The diffusion process is described by a Fokker-Planck equation (or Forward Kolmogorov equation), which in the two-dimensional case is given by ([Gar04], p.145):

$$\begin{aligned}\frac{\partial f(z,t)}{\partial t} &= -\frac{\partial}{\partial x}(a_x(z,t)f(z,t))-\frac{\partial}{\partial y}(a_y(z,t)f(z,t))+\\ &+\tfrac{1}{2}\frac{\partial^2}{\partial x\partial y}(b_{xy}(z,t)f(z,t))+\\ &+\tfrac{1}{2}\frac{\partial^2}{\partial x\partial y}(b_{yx}(z,t)f(z,t))+\\ &+\tfrac{1}{2}\frac{\partial^2}{\partial x^2}(b_x(z,t)f(z,t))+\\ &+\tfrac{1}{2}\frac{\partial^2}{\partial y^2}(b_y(z,t)f(z,t)),\end{aligned} \tag{3.5}$$

where $f(z,t) = P(Z(t) = z|Z(t_0) = z_0)$ with $Z(t) = (Z_x(t), Z_y(t))$ denoting the two-dimensional diffusion, and with the values of the coefficients given by (3.1), (3.2), and (3.3).

Our goal is to find a stationary distribution given by $f(z) = \lim_{t\to+\infty} f(z,t)$. Hence, $f(z)$ is a unique solution of (3.5) when $t \to +\infty$, in which case it holds $\frac{\partial f(z,t)}{\partial t} = 0$.

To fully describe a diffusion process we also need to know the boundary conditions. Here we have periodic boundary conditions (see [Gar04], p. 119 and p. 121) given by:

$$f(x,1)=f(x,0),\ f(1,y)=f(0,y);\ J_x(x,1)=J_x(x,0),\ J_y(1,y)=J_y(0,y), \tag{3.6}$$

where

$$J_x(x,y)=-0.5\frac{\partial}{\partial x}(b(x,y)f(x,y)),\ J_y(x,y)=-0.5\frac{\partial}{\partial y}(b(x,y)f(x,y)) \tag{3.7}$$

The unique solution of (3.5) with the conditions (3.6) is given by (3.4).

□

From Lemma 3.1 follows the following interesting property of the H-model.

Property 3.1 *Node densities λ_l and λ_h in regions A_l and A_h, respectively, relate as $\lambda_l = \gamma\lambda_h$ ($\gamma = \sigma_h^2/\sigma_l^2 > 1$), i.e., the density of nodes is γ times larger inside the disks than outside of them. Thus,* the heterogeneous speed gives rise to the heterogeneous node density that is inversely proportional to the speed.

Let us now discuss the relationship between the connectivity of nodes and the parameters of our model. The regime of interest for our model is when the node density inside the disks is high enough for clusters to form, and when the node density outside the disks is low enough for clusters not to form. We rely on results from the continuous percolation theory [MR96]. Assume a Poisson point process with density λ and assume that two nodes are connected iff they are within distance r. Then, if $\lambda r^2 > (\lambda r^2)_{cr} \approx 1.43$ (where $(\lambda r^2)_{cr}$ is called the percolation

threshold), an infinite cluster appears with a positive probability. This is called the supercritical regime. In the subcritical regime $\lambda r^2 < (\lambda r^2)_{cr}$, the clusters are almost surely finite. Although these results hold for the infinite plane, they are good approximations, as long as the node density is sufficiently high ([PP96], Proposition 2).

Therefore, we choose the parameters of our model such that we are in the supercritical regime in A_l, i.e., $\lambda_l r^2 > (\lambda r^2)_{cr}$, and such that we are in the subcritical regime in A_h, i.e., $\lambda_h r^2 < (\lambda r^2)_{cr}$.

Perfect Simulation and Discrete-Time Approximation

In this subsection, we provide an algorithm allowing for a "perfect simulation". The algorithm initializes node positions to start the simulations immediately in a stationary regime, thus obviating the need for a transient warm-up phase.

We begin the simulations as follows. We need to place n nodes according to the stationary distribution given by (3.4)[5]. First, we find the initial number N_l of nodes in A_l, which is $\sim Bin(n, \pi_l)$ (and $N_h = n - N_l$). We then need to place N_l nodes uniformly at random in the union of disks. One straightforward way to achieve this is by dropping nodes randomly on the torus, and retaining only those falling into a disk, until we have enough nodes. We proceed in analogous fashion for the N_h external nodes.

After placing the nodes, we run our simulations as follows. At every time step each node moves for $\Delta = (\Delta_x, \Delta_y)$, where Δ_x and Δ_y are independent variables chosen independently for every node and for every time step. Each of these variables has a Gaussian distribution with mean 0 and variance σ_l^2 if a node is in A_l or σ_h^2 if it is in A_h.

As the H-model is a diffusion process with spatially dependent coefficient $b(z)$, our simulations will lead to a systematic error, as shown in [FG04]. The authors in [FG04] propose an elegant solution to this problem in a general case when $b(z)$ is a differentiable function. This solution consists in correcting every step length Δ, depending on the gradient of $b(z)$ during this step. As in our case $b(z)$ is a step function their solution is not straightforwardly applicable. Nevertheless, using their approach we find the correction for the step length Δ as follows.

The step length Δ has to be corrected whenever $b(z)$ has different values during this step. Figures 3.8 and 3.9 show how to make the correction. We consider only cases when $b(z)$ changes its value once during one step, because we assume Δ is small enough that the cases are rare when $b(z)$ changes its value more than once during one step.

We look at the projections of the step length Δ into x and y coordinates and hence we show how to calculate the correct step length $\Delta_{corr} = \Delta + \epsilon$ in one-dimensional case (cf. Figure 3.10).

[5]The node positions are "almost" a Poisson process with heterogeneous intensity $\lambda(z)$, except for the condition that the total number of nodes is exactly n instead of $Po(n)$.

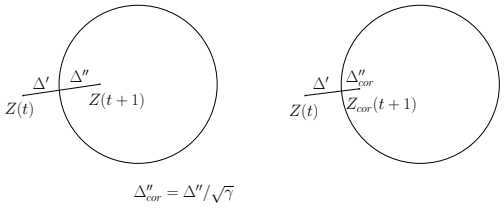


Figure 3.8: Case I - a node enters a disk.

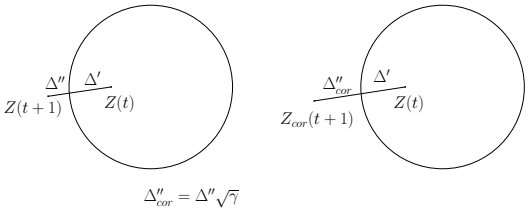


Figure 3.9: Case II - a node leaves a disk.

To calculate a correction term ϵ we start from the following equation [FG04]:

$$\epsilon = \frac{\int_{-\infty}^{\Delta} g(x)dx - \int_{-\infty}^{\Delta} g_{corr}(w)dw}{g_{corr}(\Delta)}, \tag{3.8}$$

where Δ is the step length for the case when $b(x) = \sigma_l^2$ (i.e., $b(x)$ is a constant for all x), and $g(x)$ is the distribution of the step length Δ, and $g_{corr}(w)$ is the distribution of the exact step length Δ_{corr} (when $b(x)$ is a step function).

By the normalization technique given in [Ris89], Chapter 5, we obtain:

$$g_{corr}(w) = \begin{cases} g(w) & \text{if } b(w) = \sigma_l^2 \\ \frac{\sigma_l}{\sigma_h} g(w) = \frac{1}{\sqrt{\gamma}} g(w) & \text{if } b(w) = \sigma_h^2 \end{cases} \tag{3.9}$$

We divide Δ in two parts as shown in Figure 3.10, Δ' where $b(x) = \sigma_l^2$ and Δ'' where $b(x) = \sigma_h^2$. Then, using (3.8) and (3.9) we obtain $\epsilon = -(1 - 1/\sqrt{\gamma})\Delta''$ which means that:

$$\Delta_{corr} = \Delta' + \frac{1}{\sqrt{\gamma}}\Delta''. \tag{3.10}$$

3.3.2 Validation

We now validate how well the H-model captures properties of a realistic mobility trace. We focus on the San Francisco trace only as it is the most detailed trace of

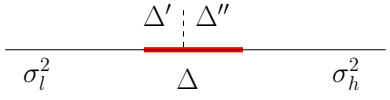


Figure 3.10: One-dimensional case.

the three considered. We check the following properties: i) statistical equivalence of nodes' mobility, ii) relation between speed and density, iii) cluster dynamics, and iv) predictive power of the H-model for an epidemic dissemination algorithm. Here, we only outline results without going into details about data analysis.

Statistical Equivalence of Nodes

We test to what extent the nodes in the San Francisco taxi trace follow distinguishable mobility patterns, e.g., because drivers might have location preferences. This is an important consideration, because it measures the potential for a routing algorithm for partitioned networks to make judicious decisions about *which node* a packet should forward to.

We assume a null hypothesis that all the node mobility patterns are drawn from the same underlying distribution, and use a Pearson's χ^2 test to decide whether we need to reject the null hypothesis, separately for each trace. More specifically, we look at the distribution of counts of visits to a grid of square cells of equal size for a time period of one day. This time period is a conservative upper bound of the time scale of interest in a routing protocol. We simply take the true distribution as the aggregate over all nodes.

We find that most of the traces (more than 60%) are statistically indistinguishable from the overall population. We note that over longer time scales, we would reject the null hypothesis more frequently, because small differences in the underlying distributions would be amplified relative to the sampling noise. However, in most delay-tolerant applications of interest, the time scale of interest tends to be of the order of minutes to hours; over these time scales, for the mobility trace at hand, we have to assume that a majority of nodes follow similar mobility patterns.

Speed and Density Maps

Here we focus on the spatial distribution of vehicle speed and density. Figure 3.11 illustrates these distributions, where darker pixels indicate higher speed/density. For the sake of visualization, we show the density distribution in a log scale (because the spatial distribution of node density is heavy tailed). The figure does suggest that speed and density are negatively correlated, i.e., that locations of high speed see low node density, and vice versa. We conjecture that this inverse relationship may be quite universal, because the mobility of nodes may be more "constrained" by other nodes in high-density areas, thus limiting their speed. This is quite easy to see in vehicular settings (highway vs. downtown traffic jam).

Cluster Dynamics

We test how well the H-model captures the cluster dynamics over the real mobility scenario. We analyze the cluster dynamics of both real clusters observed in the San Francisco trace, as well as clusters produced in the H-model whose parameters

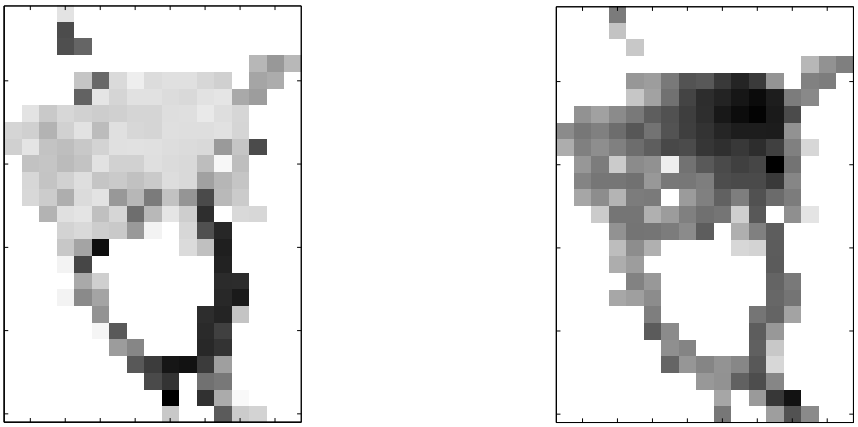


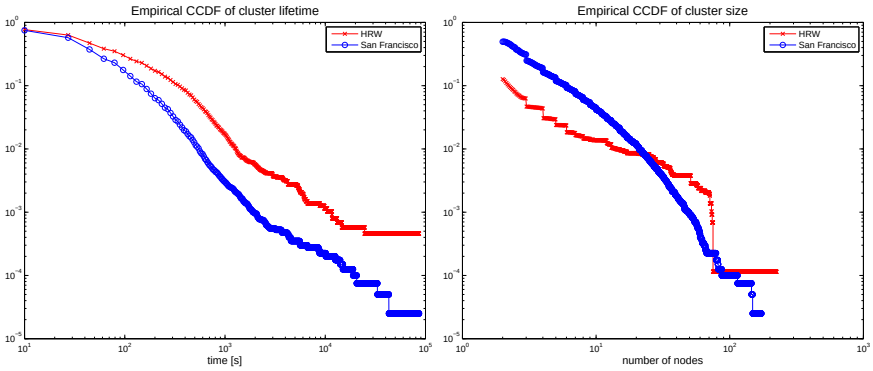
Figure 3.11: Comparison of the speed (left) and density map (right) distribution. Darker pixels indicate higher speed/density.

were inferred from the San Francisco trace [PSDG09]. For this we use the CCL algorithm explained in Section 3.2.1. We analyze the following two metrics:

- T_c: *cluster lifetime* - the time a cluster (label) is present in the network;
- N_c: *cluster size* - the average number of nodes of a cluster (label) over its lifetime.

Figure 3.12 shows the empirical complementary CDF for the cluster lifetime T_c and the average cluster size N_c. Figure 3.13 shows an evolution of the size of the largest cluster in both the model and the trace.

From these figures we observe the following. First, we see - not surprisingly - strong diurnal fluctuations in the trace but not in the model. Second, the lifetime of clusters both as produced by the model and inferred from the trace is heavy-tailed. However, the model produces more stable clusters, which in part results from the absence of diurnal fluctuations in the model, and from the sharp difference between high and low-density regions. In a real setting, the boundaries are of course more fuzzy. Third, the cluster sizes for the model and the trace are distributed rather differently. In the model, the size of each cluster tends to fluctuate in a narrow range. In the trace, there is more of a continuous spectrum of cluster sizes, again, in part, because of fluctuations during the day. Small clusters arise more frequently in the trace than in the model, which is a result of the subcritical region not being uniform as stipulated in the model.



(a) T_c: cluster lifetime (b) N_c: time-average cluster size

Figure 3.12: Comparison of cluster lifetime and cluster size in both the trace and the model.

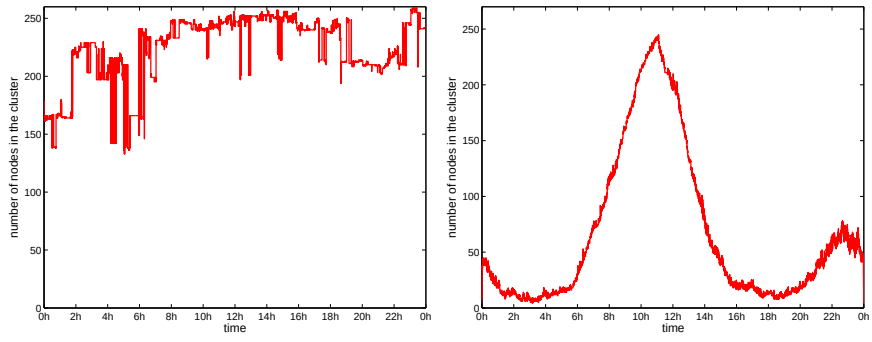
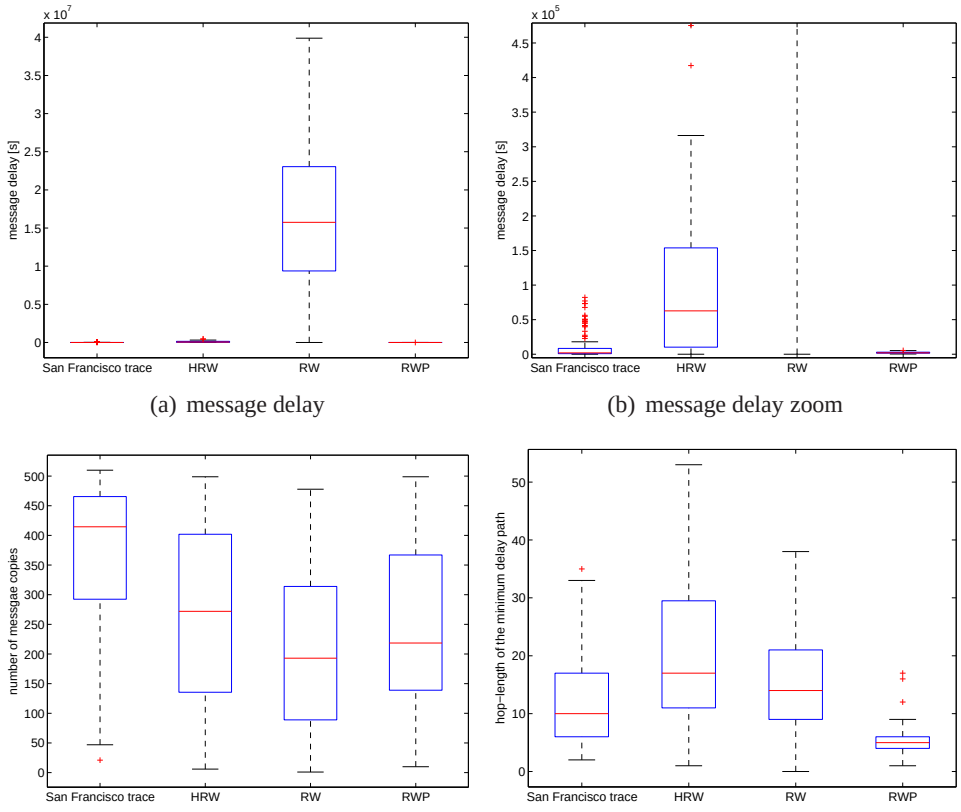


Figure 3.13: Comparison of the size evolution of the largest long lasting clusters in the model and San Francisco trace.



(a) message delay

(b) message delay zoom

(c) message copies

(d) minimum delay path

Figure 3.14: Performance evaluation of the simple epidemic dissemination protocol under different mobility processes (San Francisco trace, H, RW and RWP).

Application to Epidemic Dissemination

In this section, we evaluate the predictive power of the H-model in the context of epidemic dissemination, one of the prominent paradigms for routing in partitioned networks. We compare the H-model with the two prominent models in simulations of mobile ad-hoc networks, a (homogenous) random walk (RW) model and the random waypoint (RWP) model. For the purpose of our study, we use the following performance metrics:

- *message delay* - the time interval between sending the message by the source and receiving its copy at the destination;
- *message copies* - total number of copies made for a message in the network during the message delay time interval;
- *minimum delay path* - the hop length of the minimum delay path between the source and destination discovered by the protocol.

In Figure 3.14 we give the performance evaluation results of the epidemic protocol using a box plot for the three metrics. First, we see that a (homogeneous) RW gives a message delay that is much larger than the real delay; the H-model is still conservative, but much closer to the actual performance. This suggests that the speed at which the message spreads benefits greatly from clusters (which allows for a rapid spread of the message to other nodes), and from the low-density complement (where messages can travel quickly). The RWP model underestimates the message delay, but is closest to reality. Second, the three models predict the number of message copies fairly well. Third, the RWP model underestimate the minimum delay path. This is because the RWP model has a very small mixing time, which makes it likely that the message reaches its destination after only a small number of hops. We argue that the H-model appears to predict the performance of an epidemic dissemination protocol more accurately than other models, which suggests that clustering and spatial heterogeneity are salient features of real mobility scenarios that should not be abstracted away. However, we stress that this evaluation is not exhaustive enough to conclude that the H-model is appropriate for all scenarios of interest, and should be viewed as preliminary.

3.4 Conclusion

In this chapter we observe the following properties of realistic mobility scenarios:

- there exist CPs interspersed by low-connectivity region;
- the CPs are often not connected among each other, i.e., the network is often partitioned;
- the CPs remain stable over relatively long time-scales.

Based on these observations we devise two mobility models with the above mentioned model: the G-model and the H-model. The G-model is to be used to fit data to a model. The H-model is for modeling actual movements of nodes in the network. Hence the H-model has many less parameters than the G-model and also more realistic assumptions about nodes' connectivity. We validate the H-model by comparing its qualitative properties with a realistic mobility scenario. We find that the H-model captures well the qualitative behavior of the realistic clustered partitioned connectivity.

Chapter 4

Collaborative Routing Methods

4.1 Introduction

In the previous chapter we observe realistic mobility data sets, and we model the observed properties in the mobility with: i) a set of stable CPs interspersed by low-connectivity regions, and ii) a process that describes how nodes move between these CPs. In this and the next chapter, we design a routing algorithm for such a mobility model. In particular, in this chapter we start with the G-model as the one that gives us the essence of the problem abstracting away all complexities of the real mobility. Then, in the next chapter we consider the H-model with more realistic connectivity assumptions.

In this introductory section, we first discuss the assumptions and the main issues of the considered problem; then we explain the main ideas of our routing approach, and we review the remaining part of the chapter which presents the design and evaluation of our routing algorithm.

4.1.1 Network Model and Assumptions

The network model is given by the G-model (Section 3.2). Let us recall the notation and main assumptions. The network consists of n nodes that move independently on a CP graph $G(V, E)$ (c.f. Figure 4.1) according to an identical mobility process. The location of a node i, $X_i(t) \subset V \cup \phi$, where ϕ denotes an edge. $B_i(t)$ denotes the set of nodes that node i can communicate with (either directly or through other nodes). If $X_i(t) = v$ then $B_i(t)$ is the set of nodes at vertex v, and if $X_i(t) = \phi$ then $B_i(t) = \{i\}$.

For our purpose of the design and analysis of a routing algorithm, we focus on a Markovian process only. Specifically, we assume that each node performs a random walk on G. We will see in the next subsection that the Markovian process (a process where the future does not depend on the past, given the present), makes the routing in the G-model even more challenging.

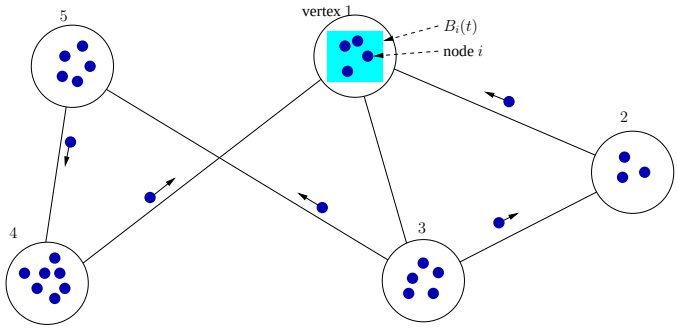


Figure 4.1: Nodes move on a graph $G(V, E)$, which describes the network topology in terms of its CPs and the ways nodes can move between them.

4.1.2 The Routing Problem Issues

We focus on the unicast routing problem, where a source node s wants to deliver a message m to a destination node d. Next we address the main issues of this problem under our network model.

Assume that s originates m at time t_0. If nodes s and d are at the same CP at time t_0, then there is an end-to-end route between s and d at this moment, and m can be delivered to node d "instantaneously" by forwarding m to the nodes along this route. Thus, the routing problem in a connected portion of the network is actually a problem for a node receiving m to decide to which of its local neighbor to forward m.

If nodes s and d are not at the same CP at time t_0 then there is no end-to-end route between s and d at this moment., and m can not be delivered to node d instantaneously. However, as nodes move they can carry m from one CP to another CP, and m can be delivered to d over time. This is so called store-carry-forwarding or mobility-assisted forwarding.

In such partitioned networks, a node receiving m needs to decide not only to which node to forward m but it also needs to decide when to forward m. Moreover, in such networks there is another useful strategy beside forwarding. It is copying m to another node instead of forwarding(i.e, a node still keeps m after sending m to another node). This strategy is beneficial because the several copies of m have a higher chance of finding d sooner than the single copy. Even in some mobility scenarios, forwarding of m could be useless, and the only possible approach could be multi-copying, as appears to be the case in the G-model. In the multi-copy approach an important decision to make for a node carrying m is when to discard m. Thus, the routing problem in the partitioned network becomes a problem of making the following decisions: i) whether to copy or to forward m, ii) when to copy/forward m, iii) to which node to copy/forward m, and iv) when to discard m. These routing decisions have to rely on opportunistic contacts and are therefore subject to the dynamics of the network topology.

The design and analysis of routing algorithms for such partitioned networks is

therefore more challenging and complex than for connected networks. The difficulties arise also because we need to take into account more important performance metrics in partitioned networks than in connected ones. In our analysis, we consider the following performance metrics:

- **transmission overhead** - the total number of transmissions of m until no more copies of m exist in the network, plus overhead due to control messages.
- **delay** - the period of time that elapses from the moment s originates m until d receives m;
- **delivery rate** - the probability that m is delivered to d.

The goal is to minimize the transmission overhead and the delay, and to have the delivery rate close to 100%. It is usually impossible to optimize all three metrics at the same time, as these are opposite requirements. So, our goal is to achieve a desirable trade-off between these metrics. In connected networks it is often enough to optimize the transmission overhead only, whereas the other two are less critical.

4.1.3 Island Hopping (IH) Overview

Next we discuss the possible routing approaches under the G-model, and then we give the main ideas behind our routing algorithm called Island Hopping (IH).

We find that only the multi-copy approach can work under our model, i.e. any forwarding of m under the our model is useless. This is because of the Markovian property of the model. An informal proof of this fact follows. Under the G-model, a node can forward m only to nodes that are at the same CP. If two nodes are at the same CP then the future movements of these two nodes are statistically equivalent, because these movements depend only on the current location. So, if a node forwards m to any other node, then the delay remains the same, but the transmission overhead increases for one transmission. Thus, any forwarding of messages is useless under the G-model with the Markovian mobility process.

Hence, we focus on multi-copy schemes only. Under the G-model, a node can make a copy of a message only to nodes located at the same CP. As the future movements of all nodes at the same CP are statistically equivalent, then it is irrelevant to which nodes at the CP a message m is copied, i.e., no statistics about the past nodes' movements can help in choosing a set of suitable relay nodes from the nodes at one CP. But, there is a value in statistics about the past mobility for making decisions of when to copy m to other nodes, i.e., at which CP to copy m. In other words, it is worth it for a node carrying m to use the past mobility information in order to decide in which CP to make copies of m, but it is not worth it to use the past mobility information for deciding to which nodes at a CP to copy m. This means that the only decision that a node can make using mobility statistics is when to make copies of m, more precisely, at which CP to make copy of m.

In our IH approach, nodes use statistics about past mobility in order to decide at which CPs to make a copy of m. In particular, the nodes collectively learn the CP topology and collectively estimate the location of the destination node d. Then the nodes exploit this knowledge to move a message through a sequence of CPs to the CP where node d is located. Each CP represents an opportunity to make new copies of m. Our idea is to make copies only in those CPs that move m closer to d in the CP graph $G(V, E)$.

The key question is how to pass a message from one CP to the next CP through nodes whose future movements are random and unpredictable, and moreover through nodes whose future movements are statistically equivalent. If the future movements of nodes were known, we could pass m to a single node at the CP that would move m in the right direction, i.e., to a CP closer to the destination in $G(V, E)$. If the future movements of nodes were statistically different, then we could choose a set of nodes at the CP that are more likely to move in the right direction. However, in our model, IH can only make a number of copies of a message at each CP, in the hope that at least one copy will move to the intended next CP. And the other copies that move to non-intended next CP are discarded. The process repeats at the next CP, until the message reaches its destination.

The key challenges are (i) not to lose the message completely, and (ii) to avoid that an unnecessarily large number of copies are generated. We observe that it is worth it to use additional statistics about the past copying of m to address these challenges. For example, if a node knows that m is already copied at a CP, then the node may decide not to make additional copies into this CP. This could be beneficial because once m reaches the next intended CP it is not useful to make more copies in the previous CPs that are thus further from d in $G(V, E)$.

For simplicity of presentation, we first describe the IH algorithm in Section 4.2 assuming that the CP graph, and node positions on the graph, are known. In Section 4.2 we also show how nodes can estimate the destination location in the CP graph. In Section 4.3, we describe *collaborative graph discovery (COGRAD)*, a distributed algorithm that infers the CP graph from each node's dynamic neighborhood set. This allows the IH algorithm to operate without any explicit clue to nodes about their location and movement. In Section 4.5, we show extensive simulation results on synthetic graphs. We show that our IH algorithm in conjunction with COGRAD results in a scheme that achieves delays of the order of much more aggressive flooding-based schemes and requires a much smaller number of copies of each message. In Section 4.6 we conclude the chapter.

4.2 Island Hopping (IH)

Island Hopping (IH) is a mobility-assisted routing algorithm in which a node makes routing decisions, i.e., when to pass a copy of a message to other nodes and when to discard it, by using the knowledge of:

1. the CP graph, and its own position in that graph, and

2. the destination's position in the CP graph.

We describe here IH assuming that this knowledge is available to the nodes. We first show the main ideas of the algorithm under the further simplifying assumption that nodes know the position in G of a message's destination. We then show how nodes can locate the destination. Last, we give a formal description of our IH algorithm.

4.2.1 Message Progression Towards a Fixed Destination

Our IH scheme uses the following three ideas, which we illustrate in Figure 4.2.

Routing a Message through a Sequence of CPs

Assume that a node i, currently located at vertex $u \in V$, has a message m with destination node d located at vertex $w \in V$. The key is for node i to decide which vertex v should be the next hop in V for message m in order to make progress towards the destination. This desired next hop v is stored in the message in the field $m.next\_hop$. We choose this next hop v as a neighboring vertex on the shortest path between vertices u and w in the CP graph.

The next move of node i is in general not yet known. If node i happens to move to the desired next hop v, then node i keeps m, and generates new copies in other nodes. If node i moves to another vertex $v' \neq v$, node i discards m.

In Figure 4.2(a), node s at vertex 1 originates a message m to node d at vertex 4. Node d makes several copies of m with $m.next\_hop := 2$. Figure 4.2(b) shows what happens when these copies move to neighboring vertices. The node with the copy that moves to vertex 3 discards m because $m.next\_hop \neq 3$, whereas the node with the copy that moves to vertex 2 makes new copies of m with $m.next\_hop := 4$. This process continues until the message reaches node d at vertex w.

At Least One Copy Moves to the Next-hop CP

If none of the copies of message m move to $m.next\_hop$, then all these copies of m will be eventually discarded, and m will be lost. To boost the probability that at least one copy of m progresses towards the next-hop vertex, we introduce a "one-hop" acknowledgement (ACK) scheme. The goal of this scheme is to piggy-back one-hop delivery information about message m through nodes moving in the reverse direction, and to generate additional copies if needed.

Assume that there exist copies of m at vertex u with $m.next\_hop = v$. Nodes at vertex u should be informed when a copy of m has reached v. When a node with m arrives at v, it broadcasts this fact to all nodes at v. If one of these nodes then moves to u, it broadcasts an ACK for m. All nodes at u can then discard m. But if a node at u holding m has not received an ACK by the time where only a small number c_1 of copies of m are left, it generates additional copies of m. This

process repeats for at most c_2 times. How many copies of m are left in a vertex can be found by a node in the following way. When copies are made at a vertex, every node remembers the set of nodes at the vertex. As the set of nodes at a vertex is proactively maintained through the vertex labeling algorithm, then a node can find out how many of m are left in the vertex by comparison with the remembered one.

In Figure 4.2(b), a node with m moves from vertex 1 to vertex 2, where it generates new copies, and broadcasts to all nodes the identity of m. In Figure 4.2(d), one of these nodes arrives at vertex 1 and broadcasts an ACK for m. Then all copies of m at vertex 2 are discarded.

Only One Copy Survives to the Next-hop CP

If more than one copy of m with $m.next\_hop = v$ moves into vertex v, then new copies of m can be generated at v several times. This could lead to an exponential increase of the number of copies. We include a mechanism to suppress additional rounds of copying. If node i moves to v, it makes new copies only if none of the nodes currently at v have seen an earlier copy of m arrive at v.

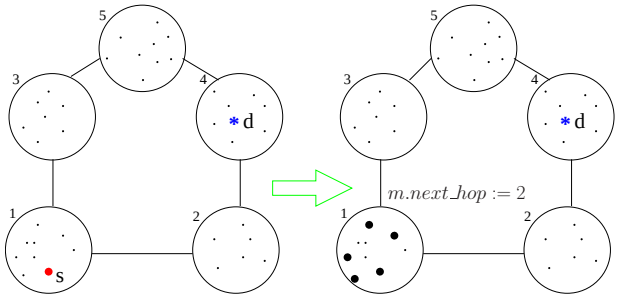
Figure 4.2(c) shows what happens when a second copy of m arrives at vertex 2 from vertex 1. Even if $m.next\_hop = 2$, the copy is discarded, because m has already been at vertex 2.

4.2.2 Dynamically Locating Destination through Last Encounter Routing

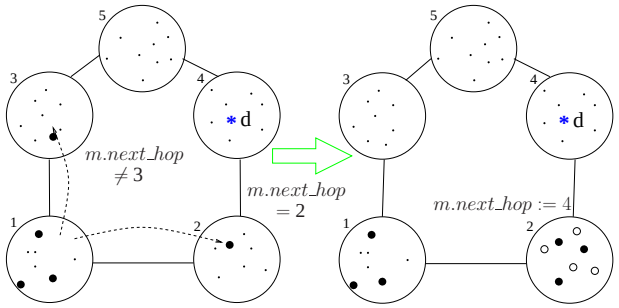
So far, we have assumed that the location of the destination is fixed and known to the message, which is unrealistic. To discover the location of the destination of a message, we cannot resort to the classical methods such as flooding, because the network is partitioned.

To solve this problem, we borrow an approach from [GV06] called Last Encounter Routing (LER), where a node maintains a Last Encounter Table (LET), with an entry for every other node. An entry consists of the time and location of its last encounter with the node. In [GV06], the location of the node is its geographic location. We adapt this to our setting, where the location of last encounter is a vertex: each node remembers for each other node the last time they were located at the same vertex, and therefore connected.

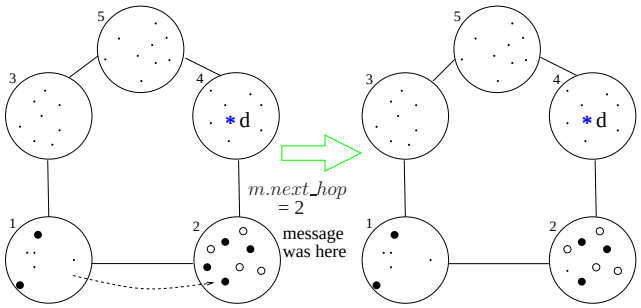
The LETs are used by a message to continually obtain more recent information about the location of the destination, as follows. Assume again that a node i at a vertex u has a message m destined for a node d. As we saw in Section 4.2.1, node i needs to determine the next-hop vertex for m. Before doing so, node i searches all nodes at u for the most recent LET entry for node d. This location is then used as an estimate of the position of node d to determine the next-hop vertex. The message remembers this estimate in a field $m.le$. As the message gets closer, it tends to find more recent information, "zeroing in" on the destination.



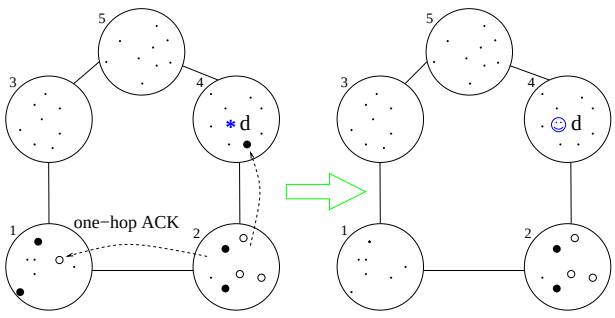
(a) Node s originates a message m to node d.



(b) Node with a copy of m moves to another vertex.



(c) Node with a copy of m moves to a vertex where m has already been observed.



(d) One-hop acknowledgment.

Figure 4.2: Island hopping - example.
Small dots - nodes without a copy of m; large dots - nodes with a copy of m; empty dots at a vertex - nodes without a copy of m, but that know that m was at this vertex.

Using the oracle knowledge about G and the destination location in G, IH tightly controls copying of a message in CPs en route, immediately killing any message that strays from the shortest path towards the destination. Using the LER approach to estimate the destination location in G, IH will make copies in CPs that stray from the shortest path, but it tends to be close to it.

4.2.3 The IH Algorithm - Formal Description

Terminology. The header of message m consists of the fields shown in Table 4.1. The $m.cid$ field is needed to differentiate between different copies of a message, and it consists of the time and location when the copy is made. The one-hop ACK of message m is denoted as $ACK(m)$.

Table 4.1: Header of a message m

$m.src$	source address
$m.dst$	destination address
$m.id$	message identifier
$m.le = \{time, location\}$	the most recent LE data for $m.dst$
$m.next\_hop$	next-hop vertex
$m.cid = \{time, location\}$	copy identifier

Node i stores messages in a *message table* M_i. While node i is at vertex v, it stores in a *visiting table* V_i the set of pairs $(m.id, m.cid)$ of messages that visit v and have $m.next\_hop = v$. As we saw above every node has a LET. The entry in the node i's LET for node j is denoted as $i.le[j]$.

The primitive broadcast(m) at vertex v transmits m to all nodes at vertex v, and copy(m) at vertex v passes a copy of m to a number n_c of nodes at vertex v. The primitive search(m) at a vertex v consists of: 1) checking if m was at v, and 2) if it was not, searching for the most recent LE data for $m.dst$ and updating the visiting tables ($V_j = V_j \cup \{(m.id, m.cid)\}$, for all $j \in B_i(t)$). Message m was at v if $(m.id, m.cid) \in V_j$, for any $j \in B_i(t)$. The result of this primitive is: (m was at v) or (m was not at v; and $best\_le = most\_recent\{m.le, j.le[m.dst]\}$, for all $j \in B_i(t)$).

The function next_hop(x, y) is the first vertex on the shortest path between the vertices x and y in the node i's view of the CP graph $G_i(V, E)$. If there are several shortest paths then one path is randomly chosen.

The IH algorithm. A node runs the IH algorithm when it originates a message, when it arrives from an edge to a vertex, when it is at a vertex and a node leaves its neighborhood, and when it receives an one-hop ACK (Algorithms 3, 4, 6, and 5, respectively).

Algorithm 3: IH: node s originates message m for node d

1 Put the following header: $m.src = s, m.id = id, m.dst = d$, , $m.next\_hop = \emptyset$, $m.le = \emptyset$, $m.cid = \{t_{now}, Y_s(t)\}$.
2 If $Y_s(t) = \phi$ /* node s at an edge */
3 $M_s = M_s \cup m$
4 Else /* node s at a vertex */
5 Node s runs Algorithm 4 (from step 3) for m
6 End if

Algorithm 4: IH: node i arrives at vertex v

1 For every $(m.id, m.cid) \in V_i$ if $m.cid.location = v$ then broadcast$(ack(m))$.
2 $V_i = \emptyset$.
3 For every $m \in M_i$
4 If $m.next\_hop = \emptyset$ then $m.next\_hop := v$.
5 If $m.dst \in B_i(t)$ then
6 Send m to node $m.dst$; $M_i = M_i \setminus m$
7 ElseIf $m.next\_hop \neq v$ then
8 $M_i = M_i \setminus m$
9 Else
10 search(m)
11 If m was at v then
12 $M_i = M_i \setminus m$
13 Else
14 $m.le := best\_le$,
$m.next\_hop$:=next_hop$(v, best_le.location)$,
copy(m).
15 End if
16 End if
17 End for

Algorithm 5: IH: node i receives $ACK(m)$

1 If $m \in M_i$ then $M_i = M_i \setminus m$.

Algorithm 6: IH: node i at vertex v sees that node j leaves vertex v

1 For every $m \in M_i$
2 If copy(m) at v happens more than c_2 times
3 $M_i = M_i \setminus m$
4 Else
5 If less than c_1 nodes at v have m then copy(m).

6 End if
7 End for

4.3 Collaborative Graph Discovery (COGRAD)

We now specify how *collaborative graph discovery (COGRAD)* infers from changes in neighborhood sets the CP graph in a distributed way, without any other signal from the environment. This is important, because it would be unrealistic to assume that the graph of CPs and the flows of mobile nodes between CPs is known a priori. Instead, we assume that the only information that nodes have available is the set of other nodes that they can reach (either directly or over multiple hops), which can be discovered in a straightforward manner (hello messages, flooding, etc.).Note that a single node would obviously be unable to find out anything about the network topology; a COGRAD protocol is necessarily collaborative.

More formally, the only input information available for COGRAD to node i at time t are:

- $B_i(t)$ - the set of neighbors of node i at time t, i.e., the set of nodes in the connected component (cluster) where node i is;
- $Y_j(t^-)$ for every $j \in B_i(t)$ - the values of labels of all nodes in the cluster where node i is at the time instant before time t (denoted as t^-).

We assume that node i knows this information at every continuous time t. As a consequence of this assumption, we assume that every node instantaneously sees any change in its neighborhood, i.e., only one node can leave or join the node's neighborhood set at time t. The goal of COGRAD is that every node i learns in a distributed way following information:

- the CP graph $G(V, E)$, and
- the current position of node i, $X_i(t)$, at all times t.

The COGRAD algorithm achieves this in two phases: vertex labeling and edge discovery. We next describe these two algorithms.

4.3.1 Vertex Labeling

The goal of this phase is to generate a label, i.e., a unique identifier, for each vertex of V, which will remain stable over time, even though nodes move in and out of each vertex. The idea is that nodes first decide whether they are currently at a vertex or at an edge; and then if they are at a vertex, they decide on a label for the vertex. Nodes make these decisions in a distributed way by looking at how their neighborhood sets change over time. Suppose that at a given time the nodes currently located at the same CP agree on a label for this vertex. Now another node

i arrives at this vertex. Node i has not received any explicit clues from the environment that it has moved, and the other nodes have not received a clue that they have not moved. However, node i's set of neighbors has changed rather markedly, whereas the other nodes' neighbor set has only seen the addition of i. These nodes can therefore decide jointly that it is likely that node i has moved, and the other nodes have not; node i therefore accepts the label of this vertex.

Under the G-model, it is easy to decide for a node whether it is at a vertex or at an edge. If the node has more than one node in its neighborhood set then it is certainly at a vertex. Under the assumption of the instantaneous neighborhood discovery primitive it is also not difficult for nodes at a vertex to decide on a label. At the startup, nodes that are at a vertex agree on a label for this vertex, e.g., by taking a random number from a large alphabet, or by taking the minimum node identifier of nodes at the vertex. This ensures an initial set of unique vertex labels. Once a vertex is labeled, then a node that comes into the vertex simply accepts an existing label. The only ambiguity there could be in this labeling process is when there is only one node left at a vertex. Then this remaining node erroneously decides that it is at an edge. Then this vertex is relabeled the next time there are two nodes at the vertex. We call this a relabeling error.

Although these errors can occur in the labeling process, this does not affect the performance of the routing algorithm if the errors occur rarely, as we will see through simulation studies in Section 4.5. In order to ensure that the relabeling errors occur rarely we set up a threshold h for a required number of nodes at a vertex for which the nodes at the vertex are allowed to associate a label with the vertex. In this way, vertices that most of the time have less than h nodes obtain labels only rarely. Thus, the goal is that vertices with a small number of nodes does not get a label at all, because there is a large probability for these vertices to stay with only one node or to become empty. In Section 4.4 we explain in more details how to set up this parameter. In Figure 4.3 we show how the vertex labeling algorithm works on an example with $h = 3$. We focus on one vertex. At the beginning, the nodes at the vertex do not have any labels. When there are at least three nodes at the vertex they obtain a random label 5 (cf. Figure 4.3(b)). When a node comes into the vertex it obtains the same label 5 (Figure 4.3(c)). The vertex looses its label 5 when only one node stays at the CP (cf. Figure 4.3(f)).

Next, we formally define the vertex labeling algorithm. Every node i maintains a variable $Y_i(t)$, which is either $Y_i(t) = \phi$ if node i thinks that it is en-route between CPs at time t or $Y_i(t) = y$ if node i thinks that it is at the CP with label y. In the former case we say that node i does not have any label, whereas in the latter case we say that node i has label y. The goal of the vertex labeling algorithm is therefore that every node i decides on $Y_i(t)$ at every time t.

From the data of $Y_j(t^-)$, $j \in B_i(t)$ node i computes: $Y(t^-) = \phi$ if every $Y_j(t^-) = \phi$ or $Y(t^-) = y$ if at least one of $Y_j(t^-) = y$. Note that in our algorithm it is not possible to have different labels among $Y_j(t^-)$. Then, node i runs Algorithm 7) with the input $(B_i(t), Y(t^-))$ to get the output $Y_i(t)$. Note that all nodes in a cluster must have the same label, because all nodes in this cluster obtain the

same information from the neighbor discovery protocol.

Algorithm 7: Vertex Labeling Algorithm - Input: $(B_i(t), Y(t^-))$**, Output:** $Y_i(t)$**.**

```
1   If |B_i(t)| = 1 then /* node i alone */
2       Y_i(t) = φ /* edge */
3   Else /* node i is at a CP */
4       If Y(t^-) = φ then /* no nodes with a label */
5           If |B_i(t)| ≥ h then /* at least h nodes at the CP /*
6               Y_i(t) =random /* a new label */
7           Else /* less than h nodes at the CP */
8               Y_i(t) = φ /* no label */
9           End if
10      Else /* nodes already have a label */
11          Y_i(t) = Y(t^-) /* keep current label */
12      End if
13  End if
```

4.3.2 Edge Discovery

As a node moves on the graph, it observes the label of every vertex it visits, as described in the previous section. If a node moves directly from vertex u with label y_u to vertex v with label y_v, then this indicates the existence of a labeled edge (y_u, y_v), and we say that the node *directly observes* this edge.

To discover the edge set E, it would be possible for each node to rely only on its own observations of the edges it traverses. However, this approach has the following drawbacks. First, if node mobility is such that a node does not visit the entire graph, then this node will never discover some parts of the graph, which can result in poor forwarding decisions. Second, even if a node moves over the entire graph, the discovery process would be rather slow, and the transient time (until every node knows most of the CP graph) would be excessively long.

We therefore would like to accelerate the dissemination of edge information to allow every node to learn the entire graph. One approach is for nodes to exchange labeled edges through a gossip protocol. This allows nodes to learn the entire CP graph more quickly.

Note however that a node's view of the CP graph may change over time because of relabeling errors. If the label of a vertex u changes from y_u to y'_u, then all edges with label y_u become obsolete. We use an aging mechanism to eliminate such obsolete edges.

More precisely, in our gossiping scheme, node i's view of the CP graph, G_i, is represented by the set of pairs (e, t_{obs}), where t_{obs} is the time when edge e was directly observed. Node i has edge e in G_i either if it has directly observed e, or if

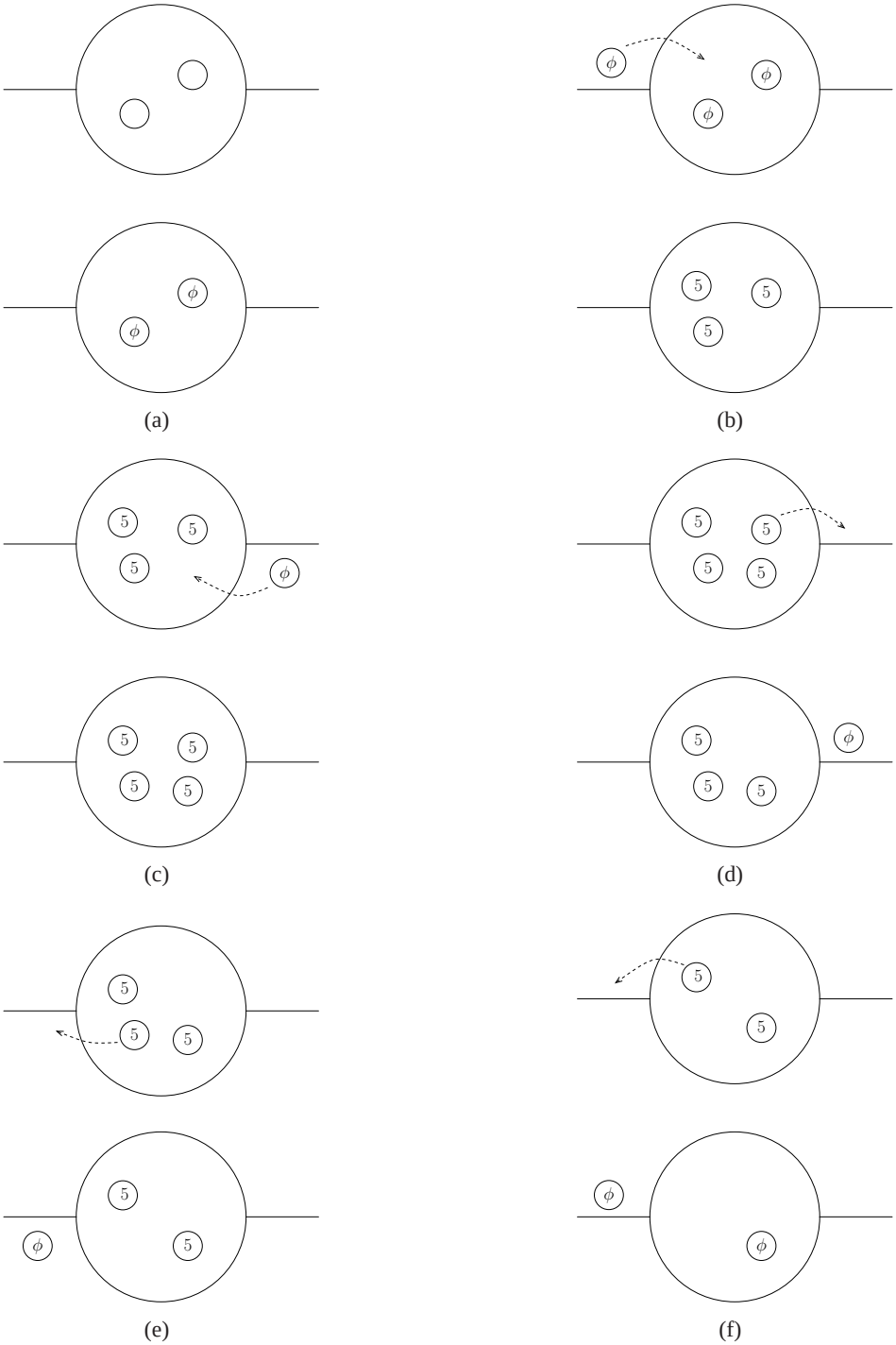


Figure 4.3: Vertex Labeling - example for $h = 3$. Evolution of a vertex over time, where each subfigure shows the vertex before labeling (up) and the vertex after labeling (down).

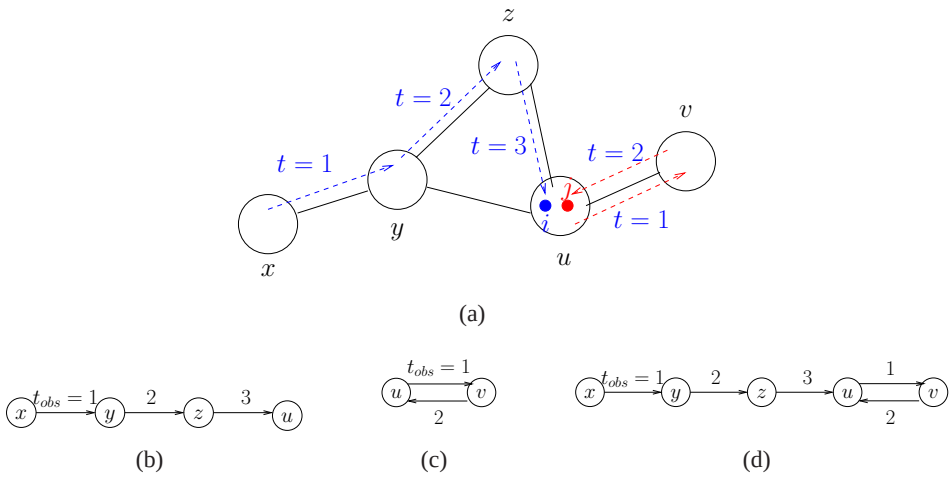


Figure 4.4: Edge Discovery - an example: a) CP graph G - Nodes i and j move on the CP graph; b) $G_i(t = 3)$, before gossiping; c) $G_j(t = 3)$, before gossiping; d) $G_i(3) = G_j(3)$, after gossiping. Note that only nodes i and j are shown, we assume there are a number of nodes in each vertex.

it has received e through gossiping from other nodes. Every node, upon arrival at a vertex, gossips a constant number (n_e) of randomly chosen entries (e, t_{obs}) from its view of the graph G_i to all nodes at this vertex. Therefore, node i updates G_i either when it directly observes an edge, or when it receives a gossip message from another node. When node i directly observes edge e at time t, it adds (e, t) to G_i and deletes the old entry for e from G_i if it exists. When node i receives (e, t_{obs}) through gossiping, it does the following. If it does not have an entry for e in G_i yet, then it adds (e, t_{obs}) to G_i. If it already has an entry for e, (e, t^i_{obs}), and if $t_{obs} > t^i_{obs}$, then it replaces (e, t^i_{obs}) in G_i with (e, t_{obs}).

In addition to the process of learning edges, a node removes an edge from its cache if the edge grows too old. More precisely, a node removes entry (e, t_{obs}) from its graph at time $t_{obs} + T_{age}$, where T_{age} is a fixed constant for all nodes. Figures 4.4 and 4.5 show examples of this process of the nodes' learning and forgetting edges of the CP graph.

4.4 Operation of IH and COGRAD

So far, we have described the IH algorithm as operating on top of an oracle that reveals to every node the CP graph G and the node's position on G. We now describe how IH operates on top of COGRAD, where the CP graph G, and each node's position on G, are obtained through COGRAD. Thus, the performance of IH depends on how successfully COGRAD infers the required knowledge.

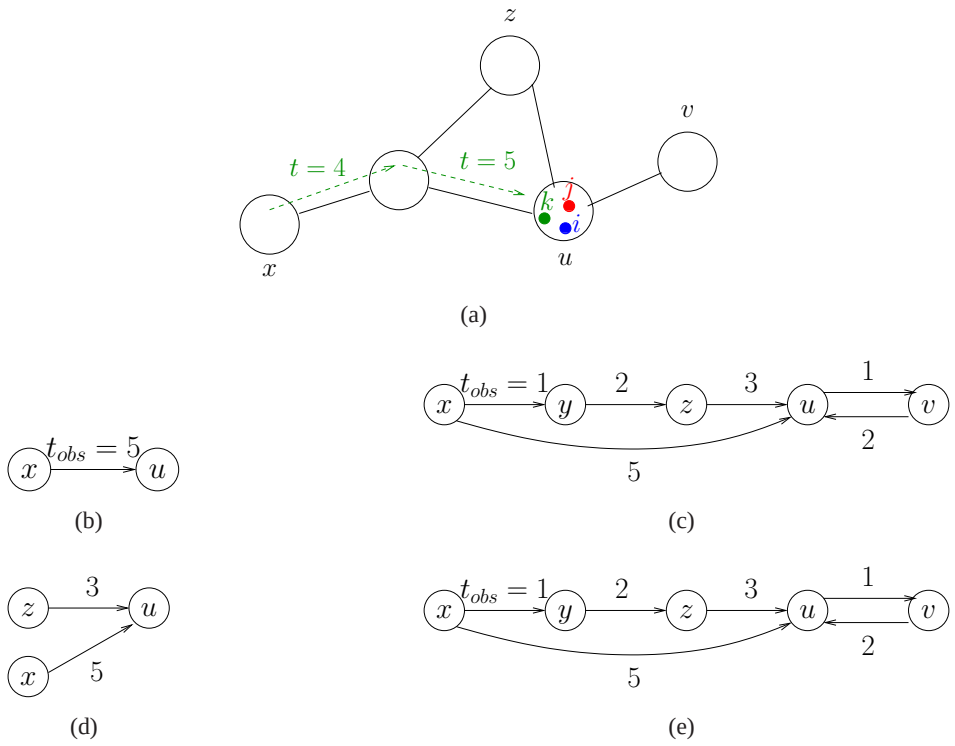


Figure 4.5: Edge Discovery - an example: a) CP graph G - Vertex y looses its label at $t = 3$, and node k moves on the CP graph; b) $G_k(t = 5)$ (note that $G_i(5) = G_j(5) = G_i(3)$) before gossiping; c) $G_i(5) = G_j(5) = G_k(5)$, after gossiping; d) $G_i(t = 7) = G_j(7) = G_k(7)$ if $T_{age} = 5$, e) $G_i(t = 7) = G_j(7) = G_k(7)$ if $T_{age} = 10$. Note that only nodes i, j and k are shown, we assume there are a number of nodes in each vertex except vertex y which is empty.

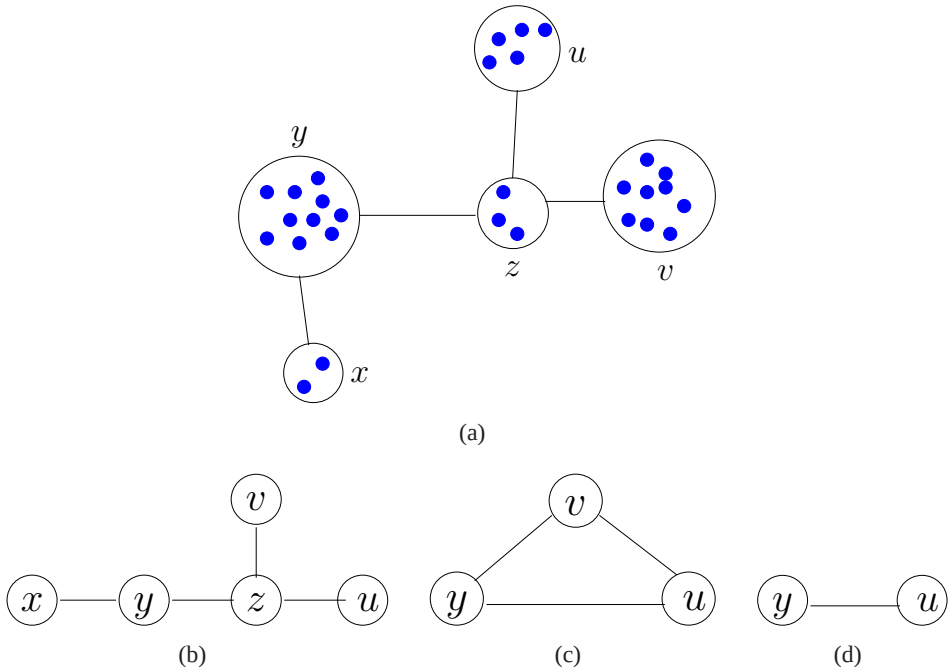


Figure 4.6: COGRAD - Impact of h: a) CP graph G b) G_i, for $h = 2$; c) G_i, for $h = 4$; d) G_i, for $h = 6$.

4.4.1 Impact of COGRAD on IH

The operation of COGRAD depends heavily on its parameters h and T_{age}. A node i's estimate G_i of G can look very differently for different values of these parameters, as we see on the examples in Figures 4.5 and 4.6. Figures 4.5(d) and 4.5(e) show G_i for different values of T_{age}. And, Figure 4.6 shows G_i for different values of h.

We may think that the optimal values of h and T_{age} are those that give an estimate closest to G. We saw that given the perfect knowledge of G and the destination location in G IH tightly controls copying of a message en route, thus immediately killing any message that strays from the shortest path towards the destination. But, this is not necessarily the best estimate for operation of IH. To explain this, let us first see what errors can appear and how they affect IH. First, there could be missing vertices or missing edges. This may lead to less efficient routing decisions in IH. That is to say, IH may make copies of a message in CPs along a much longer route than the one that would be taken if these vertices and edges were not missing. This increases both the delay and the transmission overhead. If there are many missing edges, then this may result in the inability of nodes to find a path towards the destination and as a consequence of this the message is dropped and not delivered to the destination. Second, there could be obsolete edges in G_i with labels that do not exist any more. Because of this, nodes can choose as the next

hop for a message a label that does not exist any more. In this case, the message is also dropped and not delivered to the destination. The latter error is a more serious one as it immediately leads to a drop of a message. Now let us explain why $h = 2$ is not better than $h = 4$ in the example from Figure 4.6 though G_i looks exactly the same as G for $h = 2$. This is because vertices with labels x and z have a small number of nodes, and thus there is a high probability that labels x and z will become obsolete soon. Therefore, it would be better that labels x and z are not given at all. Thus, a smart set-up of the COGRAD parameters is necessary in order that COGRAD give to IH such knowledge that makes IH perform well.

4.4.2 How to Set up COGRAD Parameters?

As we have seen an obsolete edge is more harmful for IH than a missed edge. So, we would like to set up the parameters such that it rarely happen that an obsolete edge is in a nodes' cache. But, as we want this and as we do not want that a node misses many edges of the CP graph as well, we want to find the smallest h and the largest T_{age} as possible.

We observe that there is a relation between the parameters h and T_{age} for which IH performs well. This relation allows us to set up h, given T_{age}. Then, we set up an appropriate value for T_{age} with help of simulations. We do not make an effort to calculate it analytically. One reason for this is that in Chapter 5 we design a COGRAD scheme, which can adaptively estimate this value. Another reason is that additionally we modify IH such that the modified scheme relaxes the necessity for a quick removal of obsolete edges, i.e., for a small value of T_{age}.

Next, we determine a desirable relation between h and T_{age} such that IH works well. We find first a relation between T_{age} and a desirable frequency of relabeling errors $1\tau_{err}$. Then, we find a relation between this frequency and h. Finally, we set up h as the minimum value that ensures that the frequency of relabeling errors is less than $1/\tau_{err}$.

Parameter T_{age} and Frequency of Relabeling Errors $1/\tau_{err}$

As we have seen an obsolete edge is more harmful for IH than a missed edge. An obsolete edge can appear in a node's cache either because of a change in the CP graph itself or because of errors in COGRAD. Here, we consider that the CP graph is fixed, and obsolete edges appear because of errors in COGRAD, i.e., because of the labeling process of COGRAD when a label stops to exist. Note that the same holds if a vertex in the CP graph is deleted. Thus, obsolete edges appears in nodes caches' whenever a label dies, i.e., obsolete edges appear as frequently as relabeling errors happen.

We expect that IH works well if the percentage of time that nodes have obsolete edges in their caches is much less than the time that their caches are free from obsolete edges. Let us denote the period of time during which an edge stays in the nodes' caches as a "staleness" period. Therefore, it would be desirable that the

period between two consecutive relabeling errors at a vertex is much larger than this staleness time, i.e., that τ_{err} is much larger than the staleness time. For our purposes we relax this requirement. We require that τ_{err} be at least a maximum possible staleness time. It appears that this value is good enough because we additionally modify IH to be able to handle some percentage of errors in COGRAD. The staleness time can be at most T_{age}, because edges older than T_{age} must be removed. Thus we set up h such that the desirable frequency of relabeling errors is $1/T_{age}$, i.e., we set up h such that the period between two consecutive relabeling errors at a vertex is at least T_{age} on average.

The Parameter h and Frequency of Relabeling Errors $1/\tau_{err}$

We calculate here analytically how to set up h such that we obtain a desirable frequency of relabeling errors $1/\tau_{err}$. To simplify our analysis, we look at the evolution of a single vertex. We assume that nodes arrive and leave this vertex according to a random process, where arriving nodes do not have any labels. We apply the labeling algorithm described above on this single vertex. Depending on how the number of nodes in the vertex fluctuates, the vertex is a CP with a label for some time, then it loses the label and is not a CP for some time, then it generates a new label, and so on. The vertex loses its label when there is only one node left, and after that the vertex gets a new label when the number of nodes is h. We show this in Figure 4.7, where we denote as T_{CP} the time the cluster is a CP and T_{non-CP} the time the cluster is not a CP. Hence, we want to find the minimum h such that $E[T_{CP} + T_{non-CP}] \geq \tau_{err}$.

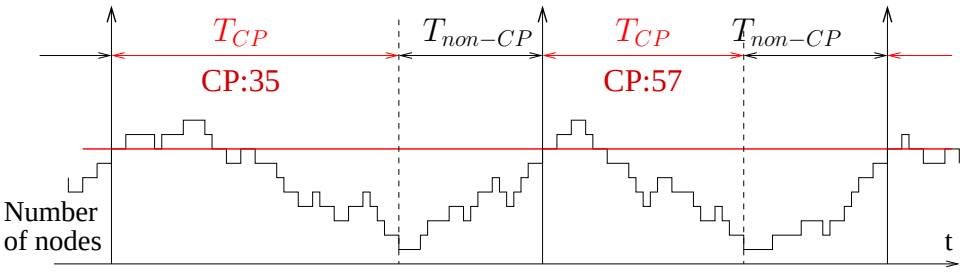


Figure 4.7: Evolution of the labeling process of a cluster.

We model the vertex with an $M/M/\infty$-queue, where nodes arrive according to a Poisson process with arrival rate λ, and every node stays for an exponentially distributed time with mean μ^{-1}. The mean number of nodes at the cluster is denoted by $\rho = \lambda/\mu$. Let the Markov process $N_t \in \{0, 1, 2, \ldots\}$ denote the number of nodes at the cluster at time t.

Let

$$D_a(b) := \inf\{t > 0 : N_t = a | N_0 = b\}$$

denote the first passage time of state a from state b. Then, the value of interest for us is an expected value of $D := T_{CP} + T_{non-CP} = D_1(h) + D_h(1)$. Therefore,

we search for the minimum h such that $E[D] \geq \tau_{err}$ for every value of λ and μ.

Using the following results [RMvdB06]:

$$E[D_1(h)] = \sum_{k=1}^{h-1} \frac{1}{\lambda} \sum_{j=k+1}^{\infty} \frac{k!}{j!} \rho^{j-k}$$

$$E[D_h(1)] = \sum_{k=1}^{h-1} \frac{1}{\lambda} \sum_{j=0}^{k} \frac{k!}{j!} \rho^{j-k},$$

we find:

$$E[D] = \frac{e^{\rho}}{\lambda} \sum_{k=1}^{h-1} \frac{k!}{\rho^k}.$$

Thus, we need to find the minimum h such that

$$F_h(\rho) = e^{\rho} \sum_{k=1}^{h-1} \frac{k!}{\rho^k} \geq \lambda \tau_{err} \text{ for every } \rho.$$

Because it is impossible to find this analytically, we plot $F_h(\rho)$ for the fixed values of $h = 10, 12, \ldots, 20$ (c.f. Figure 4.8). We see that $F_h(\rho)$ given h has one minimum over all ρ. Let us denote this minimum as F_h^{min}. Thus for any parameter settings, it holds that $E[D] \geq F_h^{min}\ \lambda$. For each value of h, we find this F_h^{min} numerically by finding ρ for which $dF_h(\rho)/d\rho = 0$ and plugging it into $F_h(\rho)$. Then, in Figure 4.9, for each h we plot the value of $E[D]$ normalized with respect to λ ($n_s = F_h^{min}\ \lambda$) that can be achieved under any parameter settings. Figure 4.9 allows us to find a proper value of h that ensures $E[D]$ is larger than the required value τ_{err}. We use this figure to properly set up the parameter h given the required $\tau_{err} = T_{hold-on}$.

Parameter T_{age}

Though it may be possible to calculate analytically this value under the G-model, we rather find an appropriate value through simulations. One reason is that in Chapter 5 we design a COGRAD scheme under more realistic assumptions, which can adaptively estimates this value. This adaptive COGRAD scheme sets up an appropriate value of T_{age} by estimating it in a distributed way. Thus this scheme adapts the value of T_{age} to the CP graph topology and the nodes' mobility. The other reason is that we additionally modify IH such that the modified scheme relaxes the necessity of a quick removal of obsolete edges.

4.4.3 IH with COGRAD

As we saw in Section 4.2, a node i located at vertex u chooses the next hop for a message m with the destination at w as the next vertex on the shortest path $u \rightarrow w$ in its view of the CP graph. We have not specified which next-hop is chosen from

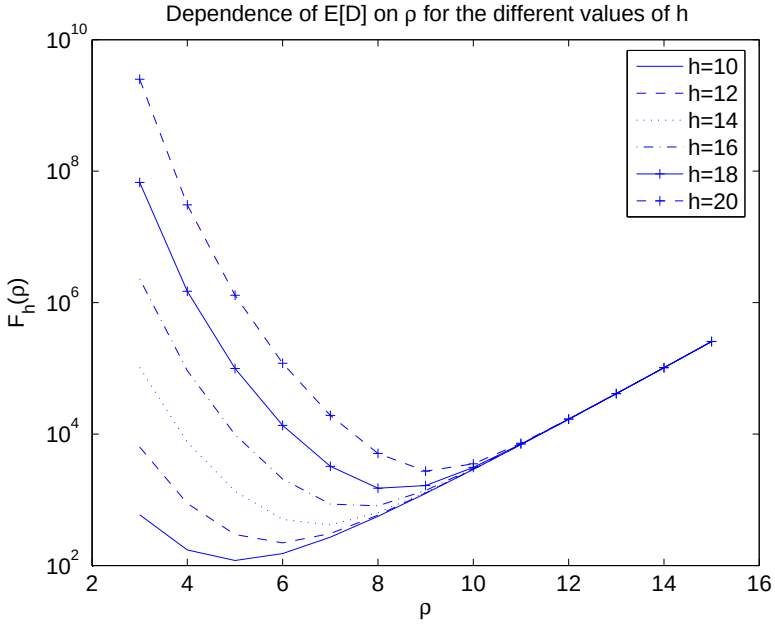


Figure 4.8: Dependence of $E[D]$ on ρ for different fixed values of h.

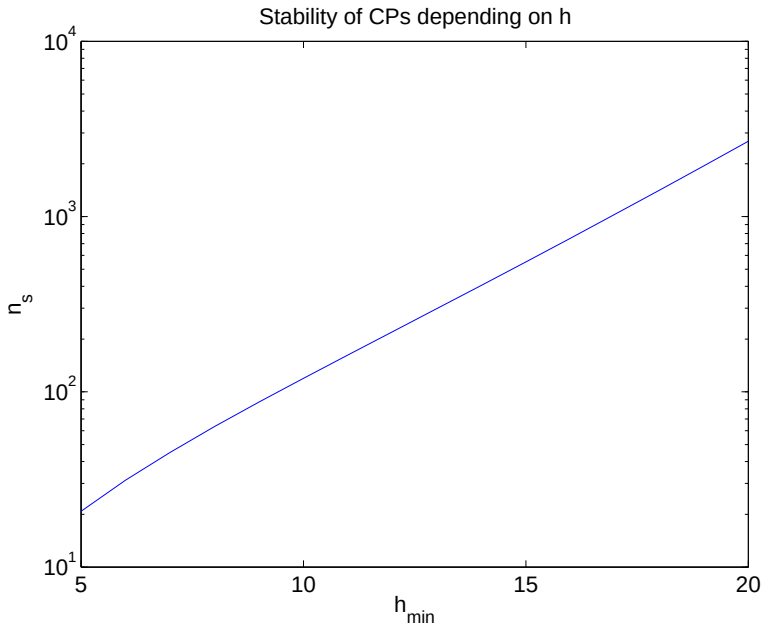


Figure 4.9: Minimum value of $E[D]$ as a function of h.

several of the shortest paths, and thus several possible next-hops, exist. In this case, we use the age of edges obtained by COGRAD in the edge discovery part to choose between these possible next-hops. We choose the next-hop with the smallest age of its incoming edge from u, i.e., the vertex v whose entry $((u, v), t)$ is most recent.

We discuss two other conditions that can arise in IH due to errors either in the underlying graph discovery algorithm or the locating of the destination. The first situation arises when node i makes the next-hop decision for a message, but cannot find a shortest path to w. This can happen when in i's view of the graph G_i, (i) w is not known, or (ii) a path $u \rightarrow w$ does not exist in G_i. The second situation arises when the destination location (obtained by the last encounter tables as described in Section 4.2) is the current node's location u, but the destination is elsewhere.

We resolve these situations as follows. The node keeps the message with setting $m.ttl = n_h$, and waits until it moves to another vertex v. There, it then tries again to find a next hop. If it does not succeed, it decrements $m.ttl$; and if it succeeds, it sets $m.ttl = n_h$. Allowing copies of a message to live for a number of hops ($n_h > 1$) until they move to the intended next CP makes IH more robust to the changes in the CP graph compared to the case where copies can live only one hop ($n_h = 1$) until they move to the next intended CP. To observe this, see the scenario shown in Figure 4.10 where an edge $56 - 95$ becomes obsolete. Assume that the node shown in vertex 56 still thinks that this edge exists. Then, this node might choose vertex 95 as the next hop for a message m. If the node discards m immediately when it moves to a vertex that is not the intended next hop, then m would be lost. But, if the node still keeps m for several hops ($n_h > 1$), then m would have a chance of reaching the desired vertex 95. Note that this modifications of IH does not incur any additional transmission overhead, however it increases memory consumption. We find this as not much of drawback, because one assumption of our work is that memory is chip and not scarce resource.

There are still some rare situations in which the IH algorithm does not succeed in delivering a message to the destination. These are: (i) when none of the message copies reach the intended next hop within the n_h traversed hops, (ii) when a CP disappears, but a node still keeps obsolete edges of this CP in the node's view of G, (iii) when a CP changes its label due to the imperfection of the labeling algorithm and does not learn edges with the new label yet but keeps obsolete edges with the old label, and (iv) if a node can not determine a next hop for a message because of the reasons explained above within the n_h traversed hops. In order to keep IH fairly simple and as the performance of our algorithm is only slightly worsened due to these rare situations as shown in Section 4.5, we do not try to make IH robust to them.

Our IH scheme of the shortest path routing in the CP graph topology optimizes the transmission overhead but does not take into account the delay. This may not be an optimal solution in some applications. For example, another approach would be to estimate the delay between two neighboring CPs, and then to find the shortest route in the CP graph with the cost of edges equal to this estimated delay. This would minimize the delay of the scheme. What is better depends on an application

and a mobility process.

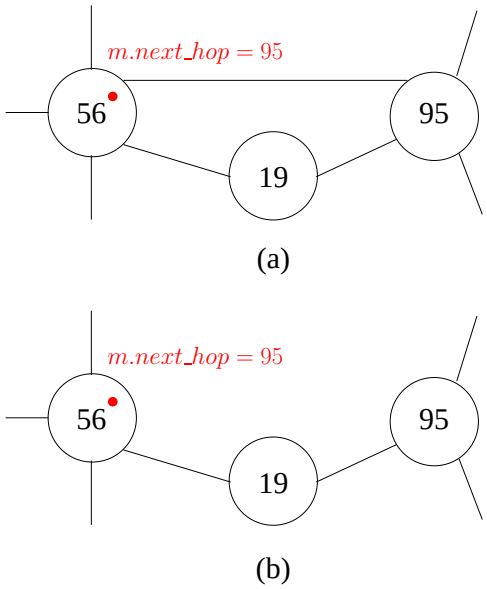


Figure 4.10: CP graph: (a) the view of the red node (b) real.

4.5 Performance Evaluation

In this section, we evaluate the performance of the IH algorithm, both in combination with an oracle that reveals the true vertex or edge identity to each node, and in combination with the COGRAD algorithm that discovers the CP graph and node locations from each node's neighborhood set $B_i(t)$, as described in Section 4.3. For this purpose, we developed a custom simulator implementing the CP graph-based mobility model, and the IH and COGRAD algorithms.

We compare our algorithm with ER [VB00] and PROPHET [LDS03b]. We also compare the delay of our algorithm with the scheme where a source transmits a message only to the destination (no routing algorithm is used). We denote this scheme as "no R". In addition, we evaluate how our algorithm depends on the various parameters.

4.5.1 Simulation Set-up

The nodes' mobility model is the G-model (described in Section 3.2). More precisely, we let $N = c|V|$ nodes perform independent random walks on a graph, with $c > 0$ a parameter controlling the mean number of nodes per vertex. Note that the random walk is the most challenging mobility process for our purposes, for the following reason. When a node performs a random walk, its future movements are

independent of the entire past. In other words, even if a node accumulates statistics about its past movements, this will not help predict the future. As such, all the nodes located at a vertex v at a given time t are statistically equivalent, and no information about the past (e.g., keeping track of when a node has last seen the destination, as in PROPHET) can be used to predict where a node will go in the future.

The location of a node is either a vertex $v \in V$ (which implies that the node can communicate with all other nodes currently located at the same vertex v), or an edge $e = (u, v) \in E$ (which implies that the node is en route from island u to island v, and is not able to communicate with any other node). Each node spends an exponentially distributed time with mean T_V at a vertex, and an exponentially distributed time T_E at an edge, where we set $T_V = 10T_E$. All the delay results we report, as well as all time scales, are normalized by setting $T_V + T_E = 1$, i.e., we normalize to unity the average speed at which a node advances from vertex to vertex.

We present simulation results for both synthetic topologies and for the city traffic topology we have inferred in Section 3.2.1. The synthetic topologies we use are:

- the 2-dimensional $k \times k$ grid, where we vary k from 3 to 9, i.e., $|V| = 9, \ldots, 81$,
- the regular random graph - a random graph where each vertex has the same degree r, with $r = 4$ and with different number of vertices $|V| = 9, \ldots, 81$,
- a diverse degree graph - a random graph with $|V| = 100$, where we divide vertices into ten groups with ten vertices of the same degree in each group and where we vary degrees of vertices from 1 to 10.

The more challenging CP graphs for our algorithm are those with the vertices having the diverse degrees. The reason is that in graphs where vertices have approximately equal degrees, the mean number of nodes per vertex is approximately equal to c for all vertices. Because of this, if c is large enough then all vertices will be well populated with nodes, and thus they will be stable CPs, which is easier for our IH algorithm to perform well. Hence, in our simulation results we include both types of graphs: those with evenly distributed degrees of vertices (the grid graph and the regular graph) and those with diversely distributed degrees of vertices (the city graph and the diverse degree graph).

So far, the only assumption we make about the set of nodes within a same CP is that each node can reach the other nodes (either because they are in direct radio range of each other, or because they can form a connected ad hoc network). However, for the purpose of simulations we assume for simplicity that nodes within a CP are directly connected. This assumption is favorable for both our algorithms and for the epidemic-based algorithms ([VB00],[LDS03a]) we compare with, because it makes it possible to transmit a message to all nodes at a CP in a single

broadcast. If nodes at a CP were not directly connected, then the broadcast function would have to be replaced by a flooding primitive.

Each simulation we report is preceded by a warm-up phase that is needed to populate the last encounter tables (LETs). The warm-up phase terminates if 80% of the node pairs have encountered each other at least once; note that this is conservative in that the LE tables are asymptotically fully populated.

Other fixed parameters in our simulations are: $c_1 = 3$ and $c_2 = 2$ in IH (defined in Section 4.2), and the maximum number of transmissions of a message allowed that one node performs in ER and PROPHET is set to 1000.

4.5.2 Performance Metrics

We use the following metrics:

- **delivery rate** - the number of messages delivered divided by the total number of messages sent by sources
- **delay** - the normalized delay for the delivered messages
- **number of transmissions per message** - the number of times a message is transmitted until there are no more copies of this message in the network.

We compute these metrics by averaging over a number of randomly chosen source-destination pairs, where for each pair a source sends a single message to its destination. In simulation results we show the mean values of these metrics with 95% confidence intervals.

Note that the number of transmissions per message includes only the transmissions of actual messages, not the control messages generated by IH and COGRAD. We justify this as follows. First, in the IH algorithm, control messages result only when a traffic message is to be transmitted. Therefore, the IH control overhead should be considered relative to the overhead to transmit messages. Usually, data messages tend to be orders of magnitude larger than control messages. This is in compliance with the proposed architecture for delay tolerant networks in [Fal03]. Therefore, the IH overhead per data message can be neglected. Second, in the COGRAD vertex labeling algorithm, control overhead accrues when a node discovers its neighborhood set $B_i(t)$, and this type of control overhead is also present in ER and PROPHET. Third, in the COGRAD edge discovery algorithm, control overhead accrues when a node arrives at a vertex and broadcasts a small number of edges. In comparison, ER and PROPHET exchange a summary vector and predictability vectors respectively, whenever a node meets a new node, which in our setting means the broadcast of these vectors per each node's arrival at a vertex. Therefore, the control overhead in our scheme is comparable to that in ER or PROPHET, and low when compared to data messages - except when the network is very lightly loaded. Hence, we consider only traffic messages in our simulation results.

4.5.3 Simulation Results

Let us recall the control parameters of our algorithm and our model. In COGRAD in vertex labeling, there is the parameter h, the threshold of the number of nodes when a new CP is formed. In COGRAD in edge discovery, there is the parameter n_e, the number of edges that each node gossips upon its arrival at a vertex, and there is the parameter T_{age}, the age threshold fixed for all nodes upon which an edge is removed from the CP graph. In IH, there is the parameter n_h, the maximum number of hops (i.e. traversed vertices) that a message can live before it reaches the desired next hop. And, in the mobility model we can vary: a graph topology G and the mean number of nodes per vertex c.

First we consider the grid 9X9 graph and the city graph with $c = 15$. For the beginning we set up $h = 2$ and $n_h = 1$. We show in Figures 4.11 and 4.12 how the performance metrics depend on n_e and T_{age}. We see that the results are not very sensitive to T_{age}, i.e., that in the broad range of T_{age} from $50(T_V + T_E)$ to $100(T_V + T_E)$ the results vary little. We also see that the good results are achieved with a relatively small number of gossiped edges ($n_e = 4$), especially in the case of the grid topology.

In the case of the taxi graph, we have a slight drop in the delivery rate (85%-95% depending on n_e). This is because the taxi graph consists of a number of vertices with small degrees (1 and 2), which results in a small mean number of nodes in a stationary regime (less than 4 and 8, respectively) and thus frequent relabeling errors occur in these vertices. This suggests that we could obtain better results if the CPs with a small mean number of nodes were actually considered edges in the CP graph.

Second, we show in Figure 4.13, the performance metrics for the grid topology as a function of the square root of the grid size k, for $c = 15$. We set $T_{age} = 70(T_V + T_E)$ and $n_e = 4$. Figure 4.13(a) shows the delivery rate. We notice that the delivery rate of the flooding-based approaches (ER, PROPHET) is essentially 100%. The delivery rate of IH drops slightly, but remains very close to 1. This slight drop-off is due to relabeling errors in COGRAD, as described earlier. We show below that this problem could be overcome with setting h and n_h appropriately.

Figure 4.13(b) shows the end-to-end delivery delay. Although the delay of IH is higher than that of flooding-based approaches (around 1.5 times higher), the figure suggests that it is only a small constant factor higher than ER/PROPHET as a function of network size.

Figure 4.13(c) shows clearly the main advantage of our scheme: it requires a significantly lower overhead per message than ER/PROPHET.

Third, we show simulations based on the inferred city graph in Section 3.2.1. In these simulations, we vary the parameter c to explore the sensitivity of IH+COGRAD to small values of c. We set $T_{age} = 70(T_V + T_E)$ and $n_e = 4$. We see in Figure 4.14(b) that small values of c are problematic; this is because vertices empty too frequently, leading to a high packet drop-rate due to relabeling errors. Setting h

and n_h appropriately can relieve this problem as shown below.

Finally, Figures 4.14(b) and 4.14(c) confirm our finding that IH+COGRAD achieves a very favorable tradeoff, with a delay close to that of flooding-based approaches that are essentially the lowest possible delay, with much lower transmission overhead, which implies that a network operating under IH+COGRAD has a capacity gain of more than an order of magnitude over ER/PROPHET for the scenarios considered here.

Fourth, we now introduce the parameters $h > 2$ and $n_h > 1$ into our scheme. Recall that the role of these parameters is to help with the problem of unstable CPs due to the relabeling errors. The larger value of h results in a longer life of labels and thus better performance of the IH algorithm. But, a too large value may result in just a small number of CPs in the network, and thus missing relatively good places to disseminate a message. This increases the delay. Also, the larger value of n_h results in the better performance of IH. But, it also increases the memory consumption. Figure 4.15 confirms this. We show the dependence of h and n_h in the diverse degree graph, which is the most challenging for IH and COGRAD of all considered topologies. We see that with $n_h = 1$ the delivery rate decreases significantly when $h < 10$. This is in compliance with our discussion about how we can set up h given T_{age} in Section 4.4.2. That is to say, by using Figure 4.8 we find that h should be around $h = 10$ given the normalized value of T_{age}, $n_s = 100$.

In Figure 4.16 we show the dependence on c in the diverse degree graph where we set $h = 15$ and $n_h = 3$. We achieve fairly good performance for such setting of parameters. We also show the results for the grid graph and the regular graph with various number of vertices in Figure 4.17 for the same parameters. And we also here achieve very good performance.

4.6 Conclusion

In this chapter we devise a novel routing algorithm for mobile partitioned networks. We evaluate it through simulations in the G models. Our IH algorithm achieves very good delay-throughput trade-off compared to the epidemic algorithm as well as PROPHET.

This favorable tradeoff is possible because our scheme tightly controls the copies of a message en route, immediately killing any message that strays from the shortest path towards the destination. In flooding-based approaches, messages diffuse throughout the network; in particular, it is difficult in these approaches to ensure that all copies of a message is discarded after one copy of the message has been delivered to the destination. This problem does not arise in IH.

To achieve this, our IH algorithm relies on the stable topology of the CPs inherently hidden in the mobility of nodes. We also devise an algorithm (called COGRAD) for discovering this CP topology in a distributed manner without any external signals from an environment.

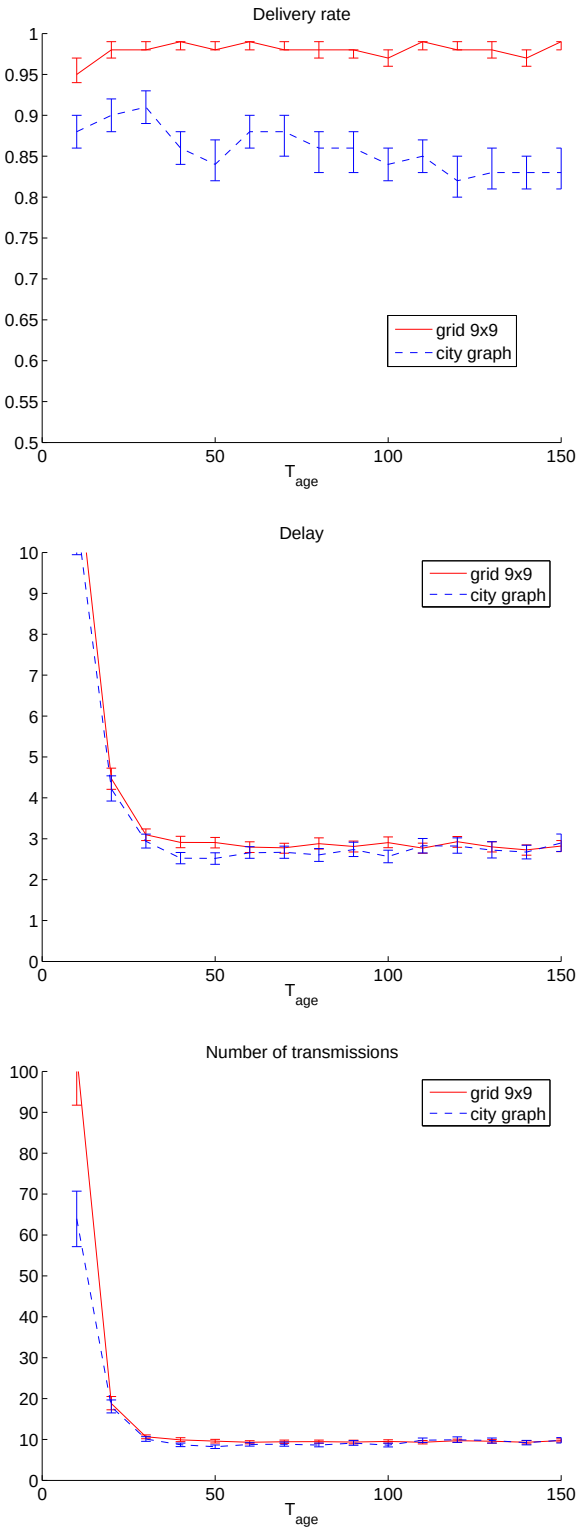


Figure 4.11: The grid 9×9 and the city CP graph; as a function of the COOGRAD parameter T_{age}; $n_e = 4$, $c = 15$. (a) Delivery rate, (b) delay, (c) number of transmissions.

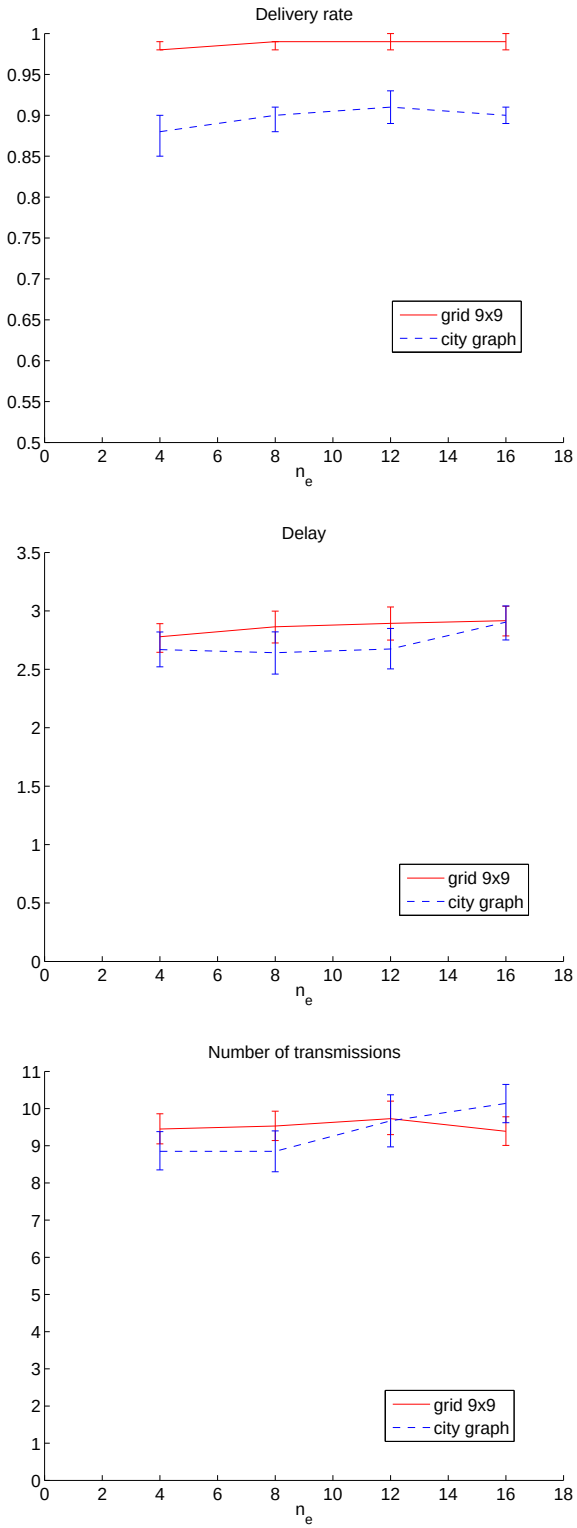


Figure 4.12: The grid 9×9 and the city CP graph; as a function of the COGRAD parameter n_e; $T_{age} = 70(T_V + T_E)$, $c = 15$. (a) Delivery rate, (b) delay, (c) number of transmissions.

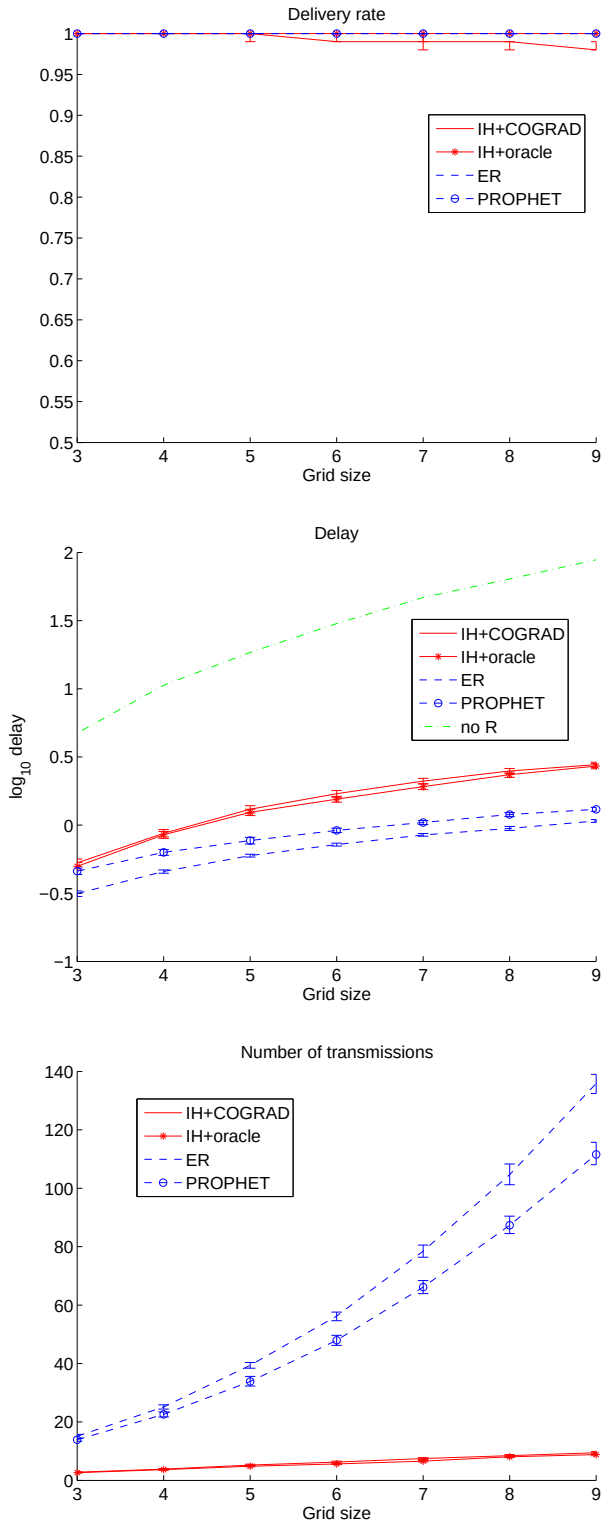


Figure 4.13: The grid topology; as a function of k (grid size $k \times k$); $c = 15$. (a) Delivery rate, (b) delay, (c) number of transmissions.

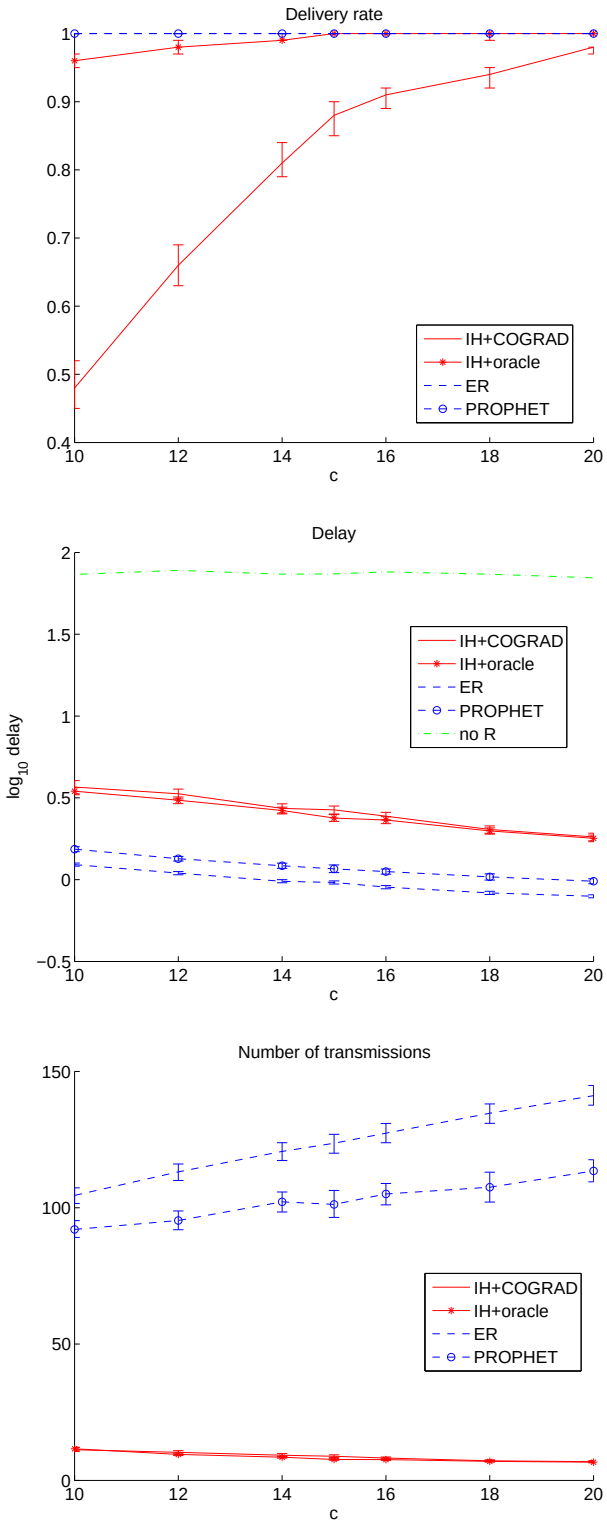


Figure 4.14: The city CP graph; as a function of c. (a) Delivery rate, (b) delay, (c) number of transmissions.

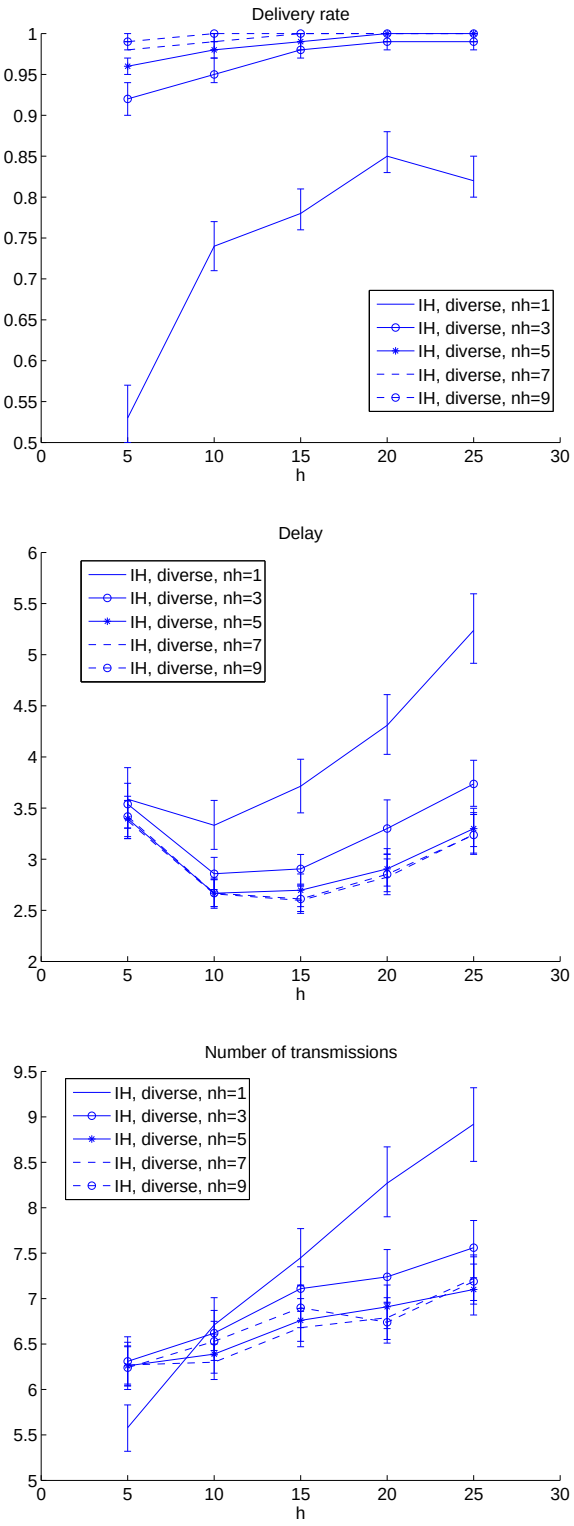


Figure 4.15: The diverse degree graph; as a function of the COOGRAD parameter h and the IH parameter n_h; ($c = 15$, $n_e = 5$). (a) Delivery rate, (b) delay, (c) number of transmissions.

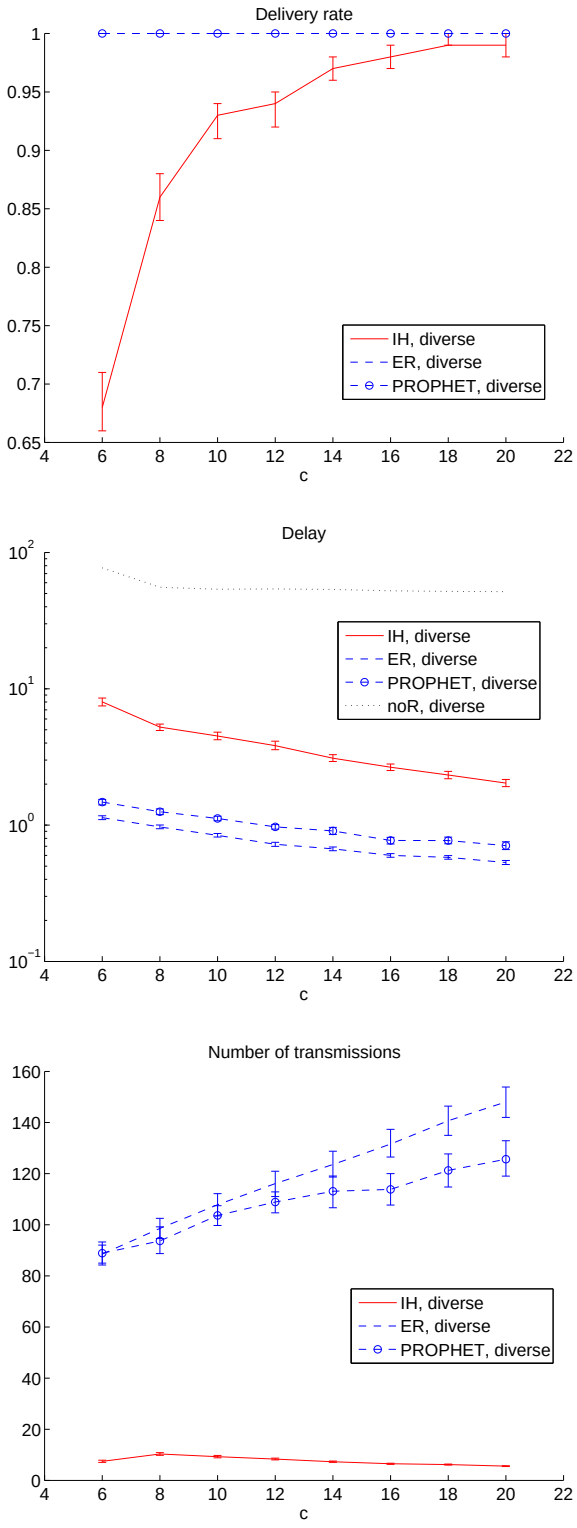


Figure 4.16: The diverse degree graph; as a function of the mean number of nodes per vertex c; ($h = 15$, $n_e = 5$, $T_{age} = 100(T_V + T_E)$, $n_h = 3$). (a) Delivery rate, (b) delay, (c) number of transmissions.

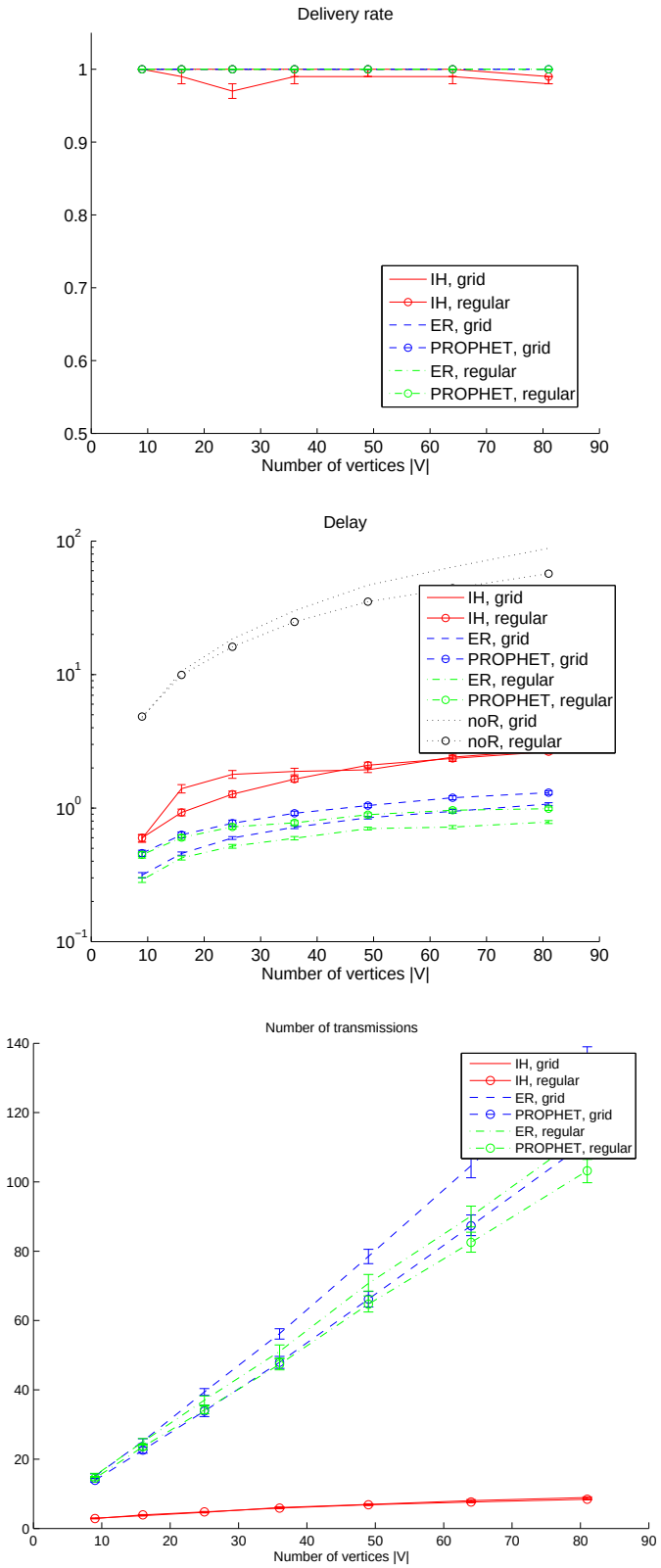


Figure 4.17: The grid and the random regular graph; as a function of the number of vertices $|V|$; ($c = 15$, $h = 15$, $n_e = 5$, $T_{age} = 100(T_V + T_E)$, $n_h = 3$). (a) Delivery rate, (b) delay, (c) number of transmissions.

Chapter 5

Relaxed Connectivity Assumptions

In the previous chapter, the design of both COGRAD and IH was driven by an abstract graph-based mobility model. In this model we make two simplifying assumptions:

- sharp connectivity between nodes at vertices and edges (all nodes in the same vertex are connected, whereas a node in transit is not connected to any other node);
- nodes can see changes in their neighbourhood instantaneously, thus we assume that only one node can leave or join a vertex within a suitable short time interval.

These assumptions allowed us to design the main principles of IH, while having a very simple COGRAD scheme.

In this chapter, we relax these assumptions. Specifically, we consider a mobility model in a geographic space with nodes connected iff they are within a certain range. For the evaluation purposes we assume the H-model, but note that we expect that the designed algorithms performs in a wider range of mobility models with the stable partitioned clusters. Also, we assume that a node sees changes in its neighbourhood periodically every Δt, instead of instantaneously. Obviously it is impossible to have an instantaneous neighbourhood discovery. There are different ways to implement a periodic neighbor discovery protocol. For example, instead of letting all nodes in a component to collect necessary information, a more efficient way would be to let only one node per component to do so. One node at a component can collect necessary information by flooding a request into the component and then collecting replies from all the nodes. We can choose this node as a first node in a component for which current information about its neighbourhood gets older than Δt. In this way, timers of nodes does not have to be synchronized.

These relaxed assumptions require the development of a more robust labeling algorithm in COGRAD, whereas Edge Discovery and IH can stay the same. Be-

side a novel labeling algorithm under the relaxed assumptions, we also design an adaptive COGRAD scheme where nodes estimate in a distributed way necessary parameters of COGRAD adaptively over time. Note that in the previous COGRAD scheme (Section 4.3), we set up these parameters assuming a concrete mobility model (the G-model with a random walk on the graph).

We evaluate IH with this adaptive COGRAD scheme using the H mobility model (Section 3.3). We show that IH performs better than the Spray and Wait (SW) [SPR05] and the Spray and Focus (SF) scheme [SPR08].

Though the adaptive COGRAD scheme performs fairly well, there are some cases where it generates unstable labels. We propose one solution to this problem called a soft labeling. In this "soft" scheme we allow to have several labels in a component contrary to the previous scheme, where only one label was allowed.

An outline of the chapter is the following. In Section 5.1 we describe how we design a labeling algorithm under the aforementioned assumptions, while in Section 5.2 we describe the adaptive COGRAD algorithm. In Section 5.3 we evaluate IH together with adaptive COGRAD by means of simulations. in Section 5.4 we discuss limitations of the labeling algorithm and we propose the soft labeling approach to overcome these limitations.

5.1 Labeling Algorithm

As we mentioned above, the relaxed connectivity assumptions require a different labeling algorithm than the one we have designed in Section 4.3.1 within the framework of the G-model. The relaxed connectivity assumption requires a new mechanism for deciding whether a node is at a CP or in transit between CPs, whereas the relaxed assumption about the instantaneous neighbourhood requires a new mechanism for the labeling of CPs. This is because the former assumption implies that between CPs there could be (transient and typically small) connected components of more than one node, and the latter assumption implies that the node's neighborhood can change by more than one node during the period of time Δt between the two neighborhood discovery updates. However, as before, we follow the principle that a label is associated with a connectivity component. I.e., the nodes connected to each other (either directly or "multi-hop" through other nodes) must have the same label, and no two components can have the same label. We denote such a labeling algorithm as *component-based*.

In the next two subsections we describe the two main ideas that govern the work of our new labeling algorithm. In Section 5.1.1 we show how nodes in a component decide on a label, and in Section 5.1.2 we show how nodes in a component decide whether the component represents a CP or an edge (for example, several nodes traveling between CPs got connected for a short period of time).

5.1.1 Majority Rules

Every node makes a new decision about its label after every neighborhood rediscovery interval Δt. We assume that the neighborhood discovery is synchronized for nodes at the same component, so that they make a decision at the same time and thus the nodes at the same component decide on the same label.

We consider that isolated nodes are in transit between CPs and thus they do not have any label. And, we consider components with at least two nodes as candidates for CPs, and thus nodes in these components get a label. There are two "majority" rules that nodes at a component use to determine a label. The first rule says the following: a label y is *possible* for component C at time t if more than the half the nodes that belong to component C at time t had label y at time $t - \Delta t$. This rule ensures that every component has a unique label, i.e., no two components can have the same label if no two components had identical labels at the previous time step $t - \Delta t$. The second rule decides on a label for C among the possible labels by choosing the one with the largest number of nodes at C at time t. If there are no possible labels for C then a new random label for C is chosen.

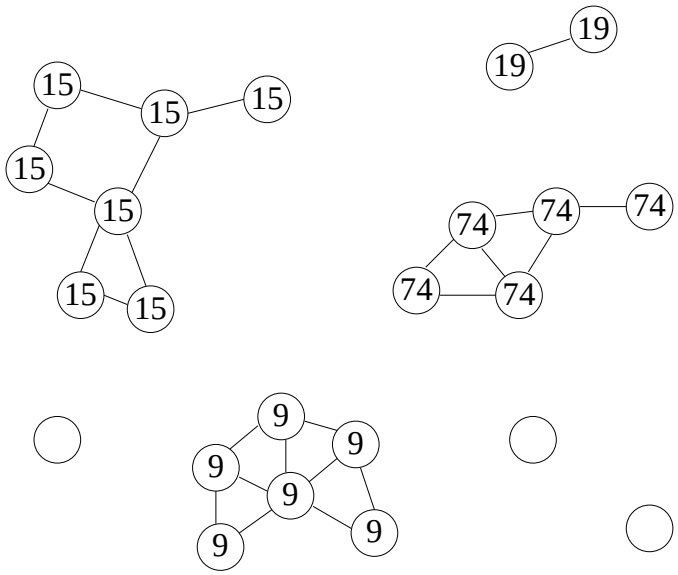


Figure 5.1: Network at time $t - \Delta t$. Nodes are shown as circles with a number in it, where the number represents a label of this node. An empty circle represents a node without any label.

We show how these rules work on the following example. Figure 5.1 shows a network at time $t - \Delta t$. Nodes are shown as circles with a number in it, where the number represents a label of this node. If a node does not have a label then there is no number in the circle. We see that at time $t - \Delta t$ there are four components with labels 15, 9, 19, and 74 and there are also several isolated nodes without any label.

Figure 5.2 shows the network at time t, before the labeling is done. We see that several nodes have moved during this period of Δt. Now, the nodes again decide about new labels. Let us explain how the nodes at C_1 decide on a label. We see that at time t at C there are 4 nodes with label 15, 3 nodes with label 74, and 2 nodes with label 9. And we see that at time $t - \Delta t$ the number of nodes with labels 15, 74 and 9 is 7, 5 and 6, respectively. Thus, using these information the nodes at C_1 decide on a label for C_1 at time t. Using the first rule they decide that labels 15 and 74 are possible. Then, using the second rule, they choose label 15. Using the same reasoning, the nodes at C_2 decide on label 9, and the nodes at C_3 decide on a new random label (e.g., 100). And, nodes that are alone loose their labels. Figure 5.3 shows the network at time t after the labeling is done.

Next, we describe the labeling algorithm more formally. Let us recall the used notation. Every node i maintains a variable $Y_i(t)$, which is either $Y_i(t) = \phi$ if node i thinks that it is in transit between CPs at time t or $Y_i(t) = y$ if node i thinks that it is at the CP with label y. The goal of a vertex labeling algorithm is therefore that every node i decides on $Y_i(t)$ at every time t.

We assume that the input information available to node i at time t is:

- $B_i(t - \Delta t)$ and $B_i(t)$ - the set of neighbors of node i (i.e., the set of nodes in the component where node i is) at times $t - \Delta t$ and t, respectively;
- $(Y_j(t - \Delta t), |B_j(t - \Delta t)|)$ for every $j \in B_i(t)$ - the values of labels and the sizes of labels of the nodes at the component where node i is at time $t - \Delta t$.

Note that actually only one node at each component needs to collect these information, which is easy to do by using the neighborhood discovery procedure. Then this node decides on a label, and then it broadcasts the label to other nodes in the component.

Now, after we defined which information is available to node i that needs to make a labeling decision, let us see how node i determines its new label $Y_i(t)$ at time t. If $B_i(t) = B_i(t - \Delta t)$ (the neighborhood set of node i did not change) then node i decides on its previous label (i.e., $Y_i(t) = Y_i(t - \Delta t)$). If $B_i(t) \neq B_i(t - \Delta t)$ (the neighborhood set of node i changed) then node i recomputes the label as follows. From the data of $Y_j(t - \Delta t)$, $j \in B_i(t)$ node i first computes the set of tuples $L_i(t) = \{(y, b^y(t), c^y(t - \Delta t))\}$, where $b^y(t)$ is the number of nodes in $B_i(t)$ that had label y at time $t - \Delta t$ and $c^y(t - \Delta t)$ is the total number of nodes that had label y at time $t - \Delta t$. If none of nodes in $B_i(t)$ have a label then $L_i(t) = \emptyset$. Then, node i runs Algorithm 8) with the input $(B_i(t), L_i(t))$ to get the output $Y_i(t)$. Note that all nodes in a component must have the same label, hence node i broadcasts $Y_i(t)$ to all nodes in the component.

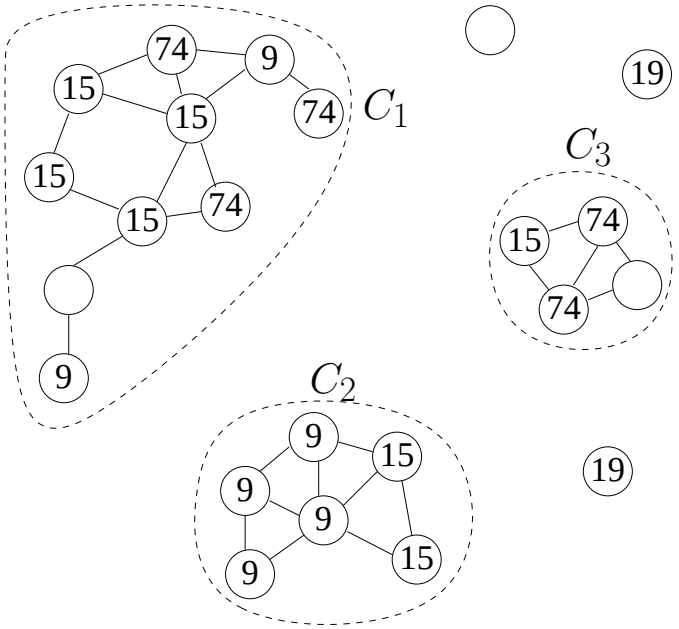


Figure 5.2: Network at time t before labeling. Nodes have moved during the period Δt.

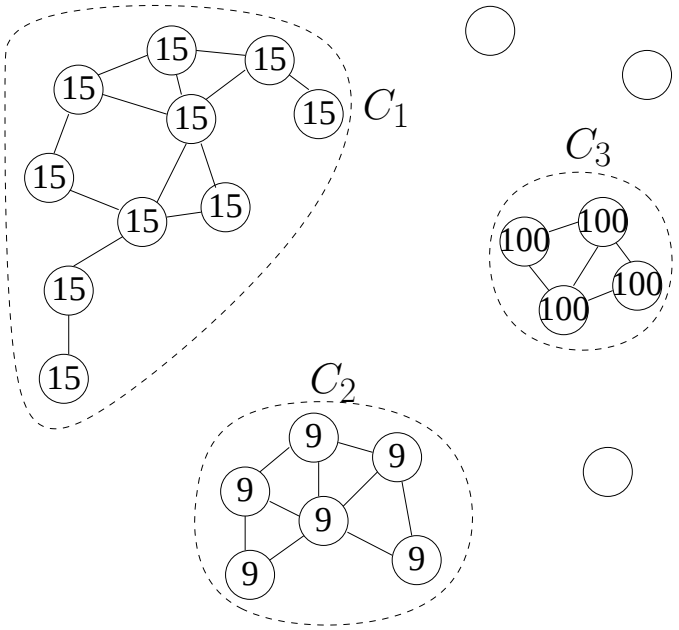


Figure 5.3: Network at time t after labeling. Nodes use two majority rules to decide on their new labels.

Algorithm 8: Component-based Labeling
Input: $(B_i(t), L_i(t) = \{(y, b^y(t), c^y(t - \Delta t))\})$
Output: $Y_i(t)$**.**

1 $Y_i(t) = \phi$
2 While $L_i(t) \neq \emptyset$
3 $z = \text{argmax}_y\, b^y(t)$
4 If $b^z(t) > c^z(t - \Delta t)/2$ then
5 $Y_i(t) := z$ /* node loses label y */
6 Break /*stop while loop */
7 Else
8 $L_i(t) := L_i(t) \setminus (z, b^z(t), c^z(t - \Delta t))$
9 End if
10 End while
11 If $Y_i(t) = \phi$ and $|B_i(t)| \geq 2$ then /* there are more than 2 nodes but no label*/
12 $Y_i(t) :=$random /* nodes assign new label */
13 End if

5.1.2 Visibility of Labels

The labeling algorithm as described in the previous subsection generates a lot of short-lived labels. This is because we assign a label as soon as a component has more than one node. In Section 4.5, we saw that if labels die frequently then this negatively impacts the performance of IH. Thus IH benefits mostly from long-lived labels.

To insure the longevity of labels, we can take an approach as before (Section 4.3.1) where a component becomes a CP only if it consists of more than h nodes. This approach gives an indirect control over life-times of CPs by appropriately setting up the parameter h. The reasoning behind this is that if a component has more nodes then the probability that the component will live longer is larger.

Here, we take another approach. We run the labeling algorithm as described in the previous subsection. Then, we track the life-time of every label. A component with a label y becomes a CP only if the life-time of label y, $\tau(y)$, is greater than a threshold value $T_{visible}$. This means that label y becomes visible to the edge discovery part as well as to IH only once $\tau(y) > T_{visible}$. We find that this approach gives us more direct control over life-times of CPs. This is because we find that the age of a component/label is strongly correlated with its residual life-time. More specifically, if a component lived for a relatively long period of time, we assume that there is a high probability that the component will be also long-lived in the future.

5.2 Adaptive COGRAD

For mobility patterns under the relaxed connectivity assumptions, our COGRAD scheme uses the vertex labeling algorithm described in Section 5.1 in conjunction with the edge discovery algorithm described before (Section 4.3.2). In this scheme we need to set two parameters, $T_{visible}$ in the vertex labeling algorithm and T_{age} in the edge discovery algorithm. Next, we show how nodes can estimate these parameters adaptively over time.

5.2.1 Estimation of Age Constant T_{age}

As we saw in the edge discovery part, a node i keeps an edge cache G_i with entries (e, t_{obs}), where t_{obs} is the time when an edge e was observed by a direct movement of some node (not necessarily node i) through edge e. Now, for the purpose of the estimation of T_{age}, node i keeps entries (e, t_{obs}, t_{update}). Here, t_{obs} is the same as previously, and t_{update} is the time when node i updated edge e. The update of edge e happens when node i learns (either directly or from other nodes) about a younger observation of e than node i currently has.

Every node estimates T_{age} for itself. The idea is that node i estimates how old on average edges in G_i become before fresher updates are received, and then that node i uses this value as an estimate of T_{age} in order to discard outdated edges in G_i. More precisely, node i calculates the mean value m and the standard deviation σ of $t_{update} - t_{obs}$ over all edges in G_i. Then, node i sets $T_{age} = m + 3\sigma$. This is one of the ways in statistics used to detect outliers in data [BG05]. Initially, each node sets $T_{age} = \infty$ and each node performs the above described calculation of T_{age} every time-step Δt until a concrete value is obtained. Once a node learns a concrete value for T_{age}, the node makes the calculations again after the estimated T_{age} time, and the procedure repeats.

5.2.2 Estimation of Visibility Threshold $T_{visible}$

As we saw, short-lived labels are not useful for IH. Therefore, we incorporate a mechanism in COGRAD to filter labels and to make visible to IH only those with long expected lifetime. The parameter that controls visibility of labels is $T_{visible}$. A larger $T_{visible}$ results in longer-lived CPs. On the other hand, too large value of $T_{visible}$ may worsen the performance of IH, because valid CPs may be missing and thus many useful opportunities for IH to copy messages are lost. So, the question is what value of $T_{visible}$ is large enough to provide enough good performance of IH.

To answer this question, let us analyse when a label is harmful for IH. When a component with a label disappears this label will remain in the node caches until it becomes older than T_{age}. During this "staleness" time the nodes have a wrong view of the existing CP graph which can lead to wrong routing decisions that negatively impact the performance of IH.

A measure of the usufulness of a label for IH is therefore the percentage of the time that the label is not stale. Hence, ideally the life-time of CPs should be much longer of the staleness time, and thus $T_{visible}$ should be much longer than the staleness time. But, it appears that it is not necessary to make so strong requirement. This is because IH is robust to some extent to errors in COGRAD. So, we try with a weaker condition, we want to ensure that the life-time of labels is at least a maximum possible staleness time. Since the maximum staleness time for any label is T_{age}, we set-up $T_{visible} = T_{age}$.

5.3 IH with Adaptive COGRAD under the H-model

In this section we evaluate our IH scheme together with the self-adaptive COGRAD under the H mobility model. We use the same performance metrics as before: delivery rate, number of transmissions, and delay. We compare it with Epidemic Routing (ER), Spray and Wait (SW), and Spray and Focus (SF).

5.3.1 The IH Scheme

We use the IH scheme described in Section 4.2 with a simple modification. Instead of discarding messages as in the previous scheme, nodes put them in a ”wait for destination” state, where nodes can copy messages only to their destinations. Thus, a node carrying a message discards it only if the node meets the destination of the message.

We make this modification in IH in order to have a fair comparison of our algorithm with others we compare with. Other algorithms also uses the same principle of discarding messages as in this modified version of IH. This modification increases the delivery rate of IH and make it the same as in other algorithms, but it requires larger memory storage. We consider a memory to not be a critical resource, and thus we do not compare algorithms according to memory requirements. Because of this we find that the more fair comparison for us is to use the described modified IH scheme.

Our IH scheme relies on broadcast and unicast primitives within a component. The task of the broadcast primitive is to deliver a message to all nodes in a component, and the task of the unicast primitive is to deliver a message from one node to another one within a component. For the purpose of our simulations, we choose a simple counter-based broadcast method ([KM05]), while instead of implementing any particular unicast routing algorithm we calculate the cost of it as the shortest path distance between two nodes in a connectivity graph.

5.3.2 Simulation Set-up

Mobility model is the H-model with the following parameters. We put circles C in a grid topology as shown in Figure 5.4, where the circles are equally spaced. We consider grids of sizes 3x3, 4x4, and 5x5.

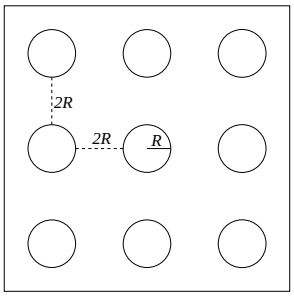


Figure 5.4: Placement of area C is deterministic in form of a grid.

The transmission range is $r_t = R/2$. We set-up σ_1 and σ_2 in such a manner that the nodes' density inside circles is $\lambda_1 = 2\lambda_0$ (λ_0 is a connectivity percolation threshold, $\lambda_0 r_t^2 \approx 1.43$), and the nodes' density outside circles is $\lambda_2 = \lambda_1/16$, i.e., $k = \sigma_1^2/\sigma_2^2 = 4$.

5.3.3 Simulation Results

In Figure 5.5 and Figure 5.6, we show the number of transmissions per message and the delay, respectively, for IH, ER and SW. The figures show results for the 3x3, 4x4, and 5x5 grid topology. The number of transmissions per message in the SW algorithm is equal to a parameter of the algorithm called the number of tokens that we need initially to set-up. We compare IH with SW by comparing the delay, while setting-up the number of tokens in SW equal to the number of transmissions per message in IH.

From these results, we see that IH has about two times larger delay than ER (which is the minimum possible one), but IH has about 10-20 times less the transmission overhead than ER. And also, the transmission overhead in IH scales much better with the grid size than in ER. Compared with SW, IH has an improvement in delay of more than 35% for all considered grid sizes.

We also compare our algorithm with SF. SF works similar to SW with the addition that nodes can also forward a message instead of copying it. Our simulations show that SF performs worse than SW. For example, for the grid 3x3 if we set the number of tokens to 10 then we get a delay of 47.57 time units, and the number of transmissions is 68.66. So, under the much larger number of transmissions per message the delay is also larger. A similar observation holds for the other grid sizes. The reason of the poor performance of SF is that any forwarding of m under the H-model does not help to improve the performance. This is due to the Markovian structure of the model as we discussed before. Hence, the expected delay in SF is not decreased by forwarding messages, and more useless transmissions are performed.

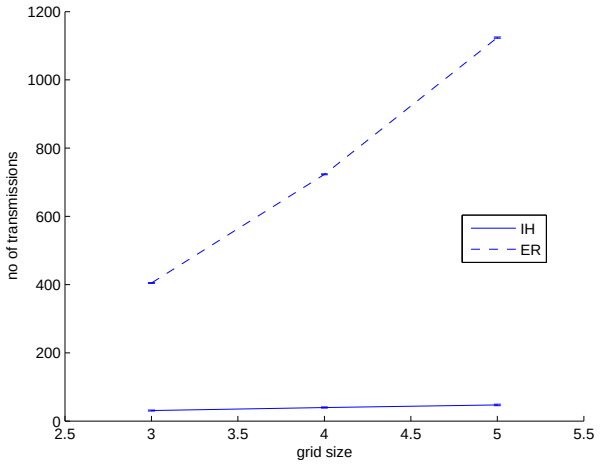


Figure 5.5: IH and ER: the number of transmissions.

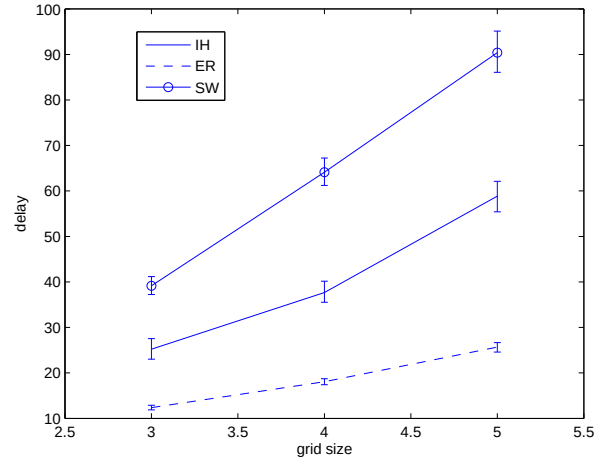


Figure 5.6: IH, ER and SW: the delay. (We set-up the number of tokens in SW approximately equal to the number of transmissions in IH.)

5.4 Soft Labeling

As we saw in the previous section, COGRAD works fairly well under the H mobility model with the disks placed into a grid. As we continue further evaluations of our algorithm under various disks placements, we found that the vertex labeling algorithm did not perform well in some cases. The critical cases are those where two disks are too close to each other. We explain why in the next subsection. To overcome these limitations, we design a soft labeling algorithm. This algorithm allows several labels to co-exist in the same connected component, contrary to the previous "component-based" labeling algorithm where only one label can exist in a component. In Section 5.4.2 we describe how the soft labeling algorithm works, and in Section 5.4.3 we confirm the advantage of the soft labeling algorithm over the component-based one.

5.4.1 Limitations of the Component-based Labeling

The component-based labeling algorithm may result in short-lived labels in the case when two disks of the H- model are relatively close to each other even under the supercritical connectivity inside the disks and the subcritical connectivity outside of the disks. An example in Figure 5.7 shows this. We see that initially the nodes in the disks form two disconnected components that thus have different labels 15 and 9. Then, when the nodes in the discs get connected, they all have now the same label 15. When the nodes in the disks get disconnected again, one of the disks get a new label 19. Note that in this example relabeling errors happen in the both disks. Depending on the distance between the disks, the transmission range and the node densities, this connection and disconnection of the nodes in the two disks may happen often, thus resulting in short-lived labels.

We noticed that the same situation happen in the simulation scenario considered in the previous section even under the subcritical regime of λ_2 if λ_2 becomes large enough. As we will see later the soft labeling algorithm helps to overcome this problem.

5.4.2 The Soft Labeling Algorithm

Algorithm 9 formally describes how the soft labeling algorithm works at a node i. Nodes run the algorithm every Δt as before, after every new neighborhood discovery. For each component, the nodes first calculates the set of possible labels Y_p. A label y is possible for a component if the majority of the nodes that were at CP y at the previous time step $t - \Delta t$ is still in the component at time t. Then, every node in the component chooses one label from Y_p. A node i chooses label $z \in Y_p$ if the majority of nodes in node i's local neighborhood $N_i(t)$) had label z at time $t - \Delta t$.

Figure 5.8 shows how the soft labeling algorithm works in the example scenario shown in Figure 5.7. We see that it succeeds to maintain the initial labels of the

disks regardless of the disks' connection and disconnection. Thus, in this critical scenario the soft labeling creates longer-lived labels compared to the component-based labeling.

Algorithm 9: Soft Labeling
Input: $(B_i(t), L_i(t) = \{(y, b^y(t^-), c^y(t)\}, N_i(t))$
Output: $Y_i(t)$

```
1   Y_i(t) = φ, Y_p = ∅
2   For all y in L_i(t) ≠ ∅
3       If b^y(t) > c^y(t − Δt)/2 then Y_p = Y_p ∪ {y}.
4   End For
5   For all y in Y_p ≠ ∅
6       n_y = Σ_{j∈N_i(t)} 1_{Y_j(t)=y}
7   End For
8   If Y_p ≠ ∅
9       z = argmax_y{n^y(t)|y ∈ Y_p}
10      Y_i(t) = z, τ(Y_i(t)) = τ^z(t^-) + 1)
11  Else
12      If |B_i(t)| ≥ 2 then
13          Y_i(t) =random
14      End If
15  End If
```

5.4.3 Evaluation

We compare the soft labeling algorithm (Algorithm 9) with the component-based labeling algorithm (Algorithm 8), under the H mobility model. We use a metric that shows a labeling quality inside CPs. Ideally, inside a CP (a disk in the H-model) there should be only one "real" label. Hence, we look at one disk C and we consider the average percentage of time that there is one and only one "real" label at C. So, the metric we use is:

$$Q = \frac{1}{t_{end} - t_{start}} \sum_{t=t_{start}}^{t_{end}} q(t), \tag{5.1}$$

where $q(t)$ is given by:

$$q(t) = \begin{cases} 1 & \text{if exactly one label is located at } C \\ 0 & \text{otherwise} \end{cases} . \tag{5.2}$$

The location of a label is a "center of mass" of locations of all nodes with this label.

The simulation set-up of the disks in the model is the same as in Section 5.3.2. Figure 5.9 shows the dependence of q_{in} on the parameters of the H-model λ_1 and

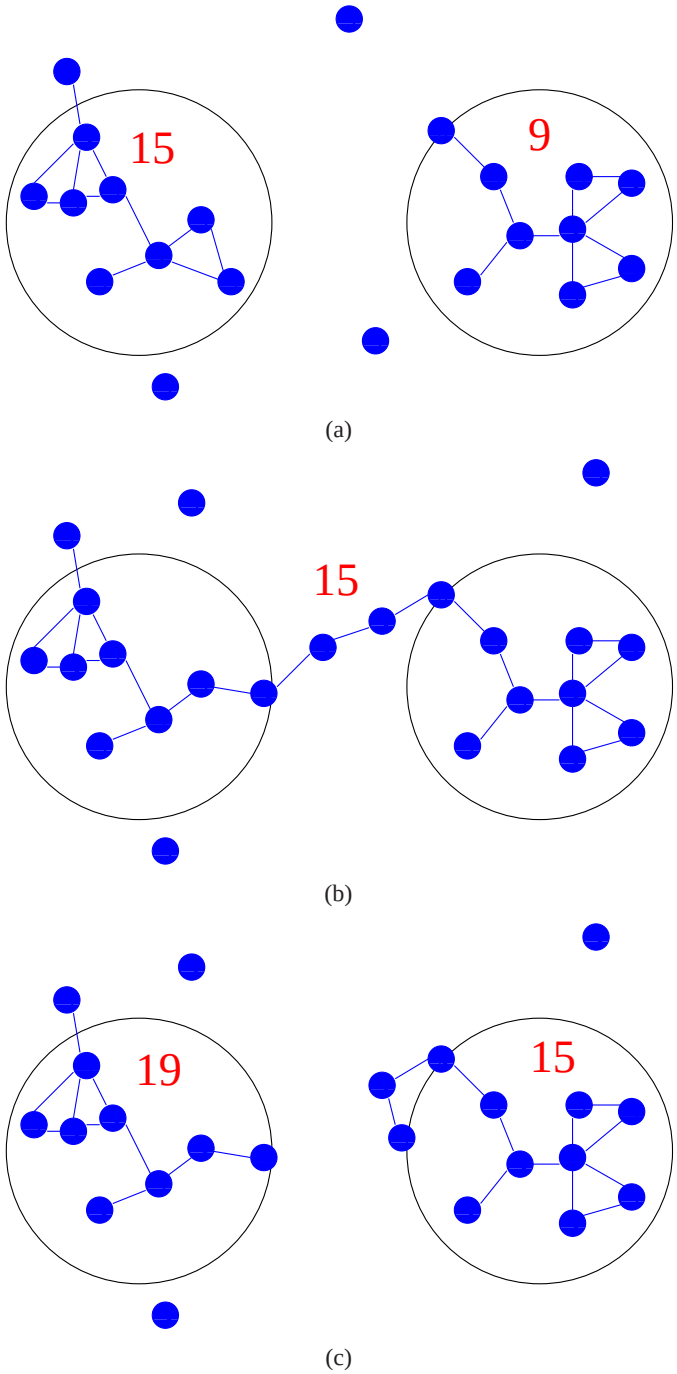


Figure 5.7: Relatively close CPs; the component-based labeling: a connection and a disconnection of two close CPs results in relabeling errors; a) two CPs are disconnected, b) the two CPs get connected, c) the two CPs get disconnected again.

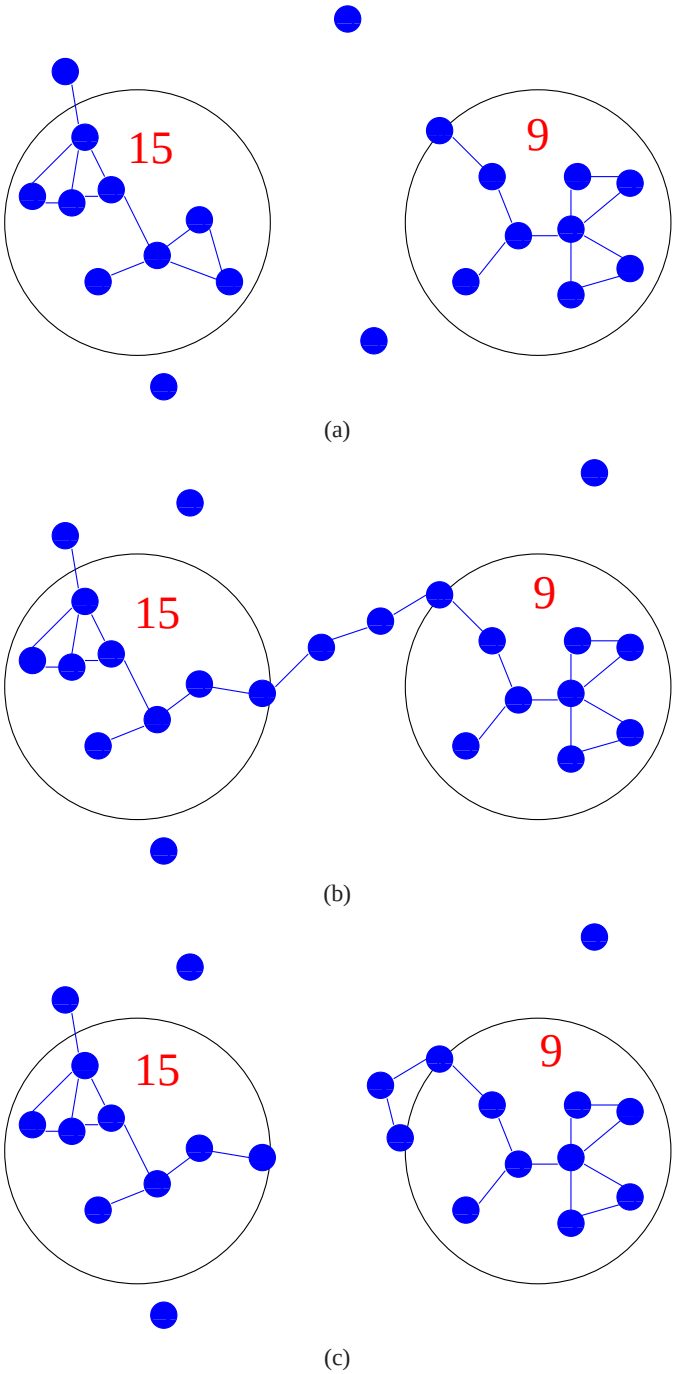
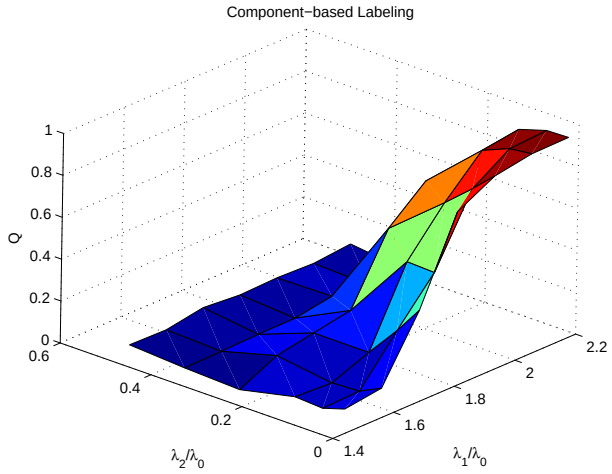
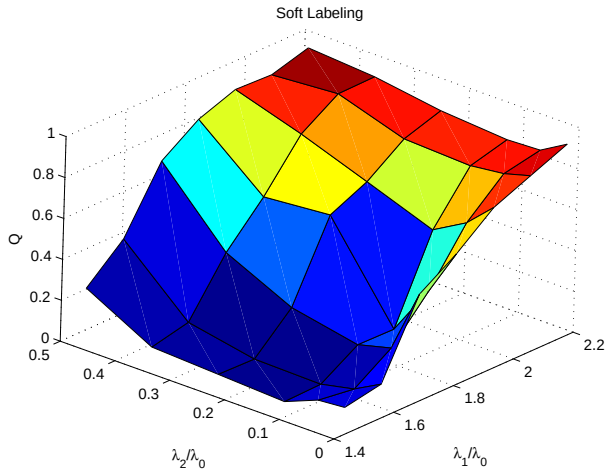


Figure 5.8: Close CPs - soft labeling: no relabeling errors; a) two neighboring CPs are disconnected, b) the two CPs get connected, c) the two CPs get disconnected again.



(a) Component-based Labeling.



(b) Soft Labeling.

Figure 5.9: Dependence of the quality of labeling inside CPs Q on the nodes' densities inside and outside CPs (λ_1 and λ_2, respectively). λ_1 and λ_2 are normalized on the percolation threshold $\lambda_0 \approx 824$.

λ_2, the nodes' densities inside and outside disks, respectively. We see that the soft labeling significantly outperforms the component-based labeling over a wider range of the parameters of the H-model. More specifically, we see that the quality of the soft labeling inside CPs is close to 1 for much larger values of λ_2 than for the component-based labeling. This indicates that the soft labeling algorithm overcomes successfully the close disks problem that appears for larger values of λ_2 as described in Section 5.4.1.

5.5 Conclusion

We design IH and COGRAD in both this and previous chapters, but under different assumptions about mobility of nodes and about discovery of nodes' neighborhoods. The assumptions in this chapter are closer to realistic conditions, thus we make further steps towards designing IH and COGRAD in realistic networks.

For the mobility of nodes, we assume the G-model in the previous chapter, while we assume the H-model in this chapter. The H-model relaxes the sharp connectivity assumption incorporated in the G-model. Beside this, also the fact that the H-model requires a small number of parameters makes the H-model more suitable than the G-model for simulations and analysis of our algorithms. For the neighborhood discovery primitive, we assume an instantaneous neighborhood discovery in the previous chapter, whereas we assume a periodic neighborhood discovery (at every rediscovery interval) in this chapter. Obviously, the latter assumption is more realistic.

Note that both the G-model and H-model represent challenging cases that DTN routing algorithms should be able to handle, where more realistic connectivity assumptions make the H-model more challenging. The key distinguishing features of these models are that (i) nodes are statistically equivalent, (ii) clustering arises as a property of the environment, rather than as a property of the nodes, and (iii) a mobility process of nodes possesses Markov property. It should be clear that the former two features would hamper any attempt to base routing decisions on *nodal statistics* (e.g., preferred locations for particular nodes, or encounter frequencies between specific pairs of nodes), which appears to be an implicit or explicit assumption built into many routing protocols. Moreover, the Markov property would hamper any attempt to use *past mobility statistics* in order to decide on a suitable subset of nodes from nodes in a CP for coping/forwarding a message to. While the last mentioned feature is true in the G-model, it is actually "close to true" in the H-model. The reason is that nodes at the same CP differ in their locations slightly, thus they are not exactly statistically equivalent given the past mobility of nodes. But, this difference is small and is practically useless to make any meaningful routing decisions about to which nodes at a CP to pass a message.

In this chapter, we show that IH and COGRAD can operate well under the challenging H-model and under a realistic assumption about the neighborhood discovery primitive. Moreover, under these challenging assumptions we design an

adaptive COGRAD algorithm that estimates in a distributed way its parameters adaptively over time. How well this COGRAD scheme operates depends mainly how well the labeling algorithm operates, i.e., how stable labels it gives. And this depends on a mobility scenario. For example, in some critical scenarios of the H-model a component-based labeling scheme used in the adaptive COGRAD algorithm is not a best choice. We show that another soft labeling scheme has more promising applications in these scenarios.

Therefore, in this dissertation we design IH and COGRAD under fairly realistic and challenging assumptions. But, it is also necessary to evaluate our algorithms under realistic data sets. For example, an open question is which labeling scheme is more suitable under these realistic data sets. In fact, we expect that under the realistic data sets there will be some situations where COGRAD will not perform well. Thus, we expect that there will be necessary to make some accommodations of COGRAD to these situations, but we expect that IH can stay the same with eventually minor modifications. Note that for the design of IH and COGRAD it was an essential to first use a model for node's mobility instead of using realistic data sets immediately. This is because the considered data may not contain stable CPs and thus it would be difficult to determine if it is a property of the data or unsuccessful operation of COGRAD. Moreover, the model gives us a possibility to evaluate how the performance of the algorithms depends on various parameters (such as node density, the number of nodes in CPs or in total, the number of CPs, etc).

Chapter 6

Conclusion

In this thesis we consider the unicast routing problem in mobile ad-hoc networks. We devise efficient routing methods by exploiting features of mobility common to many real-world applications of mobile ad-hoc networks. We exploit two features in particular, i) predictability and ii) temporally stable and spatially heterogenous distribution of nodes.

Exploiting predictability, we improve significantly the performance of LER under the RWP mobility model. Studying LER under RWP is important because the RWP model is very prominent model in mobile ad-hoc networks and the previous work on LER [GV03] showed that the LER algorithms EASE and GREASE did not perform well under RWP. We derive optimal predictors under the RWP model. The predictors optimize the mean square error between predicted and real locations under different observation sets. As observations, we use speed and location of a node, and as possible additions previous and next waypoints.

Exploiting temporally stable and spatially heterogenous node distribution, we devise Island Hopping (IH), a novel efficient routing algorithm for realistic mobile ad-hoc networks. These features of mobility appear to be present in large-scale realistic mobility traces that we investigated. We argue that this could be a common pattern in many realistic scenarios, because it is natural or constructed environments that limits the nodes' movements. The limitations imposed by the environments typically leads to heterogeneous connectivity and to network partition, where highly connected clusters are interspersed with low-connectivity regions. We model such a situation with a set of stable concentration points (CPs) characterized by high node density, and a mobility process that describes how nodes move between these islands of connectivity. We study two instances of this model: the G-model, where the CPs and the flow of nodes are abstracted as a graph, and the H-model, where nodes perform heterogeneous random walks on the plane.

Assuming the described mobility model with stable CPs, we devise the routing algorithm IH. It exploits the knowledge of the stable CP topology to make routing decisions. In this way IH achieves a very good delay-throughput trade-off compared with several other main routing approaches for mobile partitioned networks,

and moreover IH scales well with the network size. Specifically, we compare IH with Direct Forwarding (a message is forwarded only to the message's destination), Epidemic Routing (ER), PROPHET, Spray and Wait (SW), and Spray and Focus (SF). The chosen algorithms represent the main approaches in the state of the art at the time we designed IH.

For making our first steps towards the design of IH we used the G-model which was useful to devise the main principles of IH. Then we use the H-model, as a more realistic model regarding connectivity assumptions. Note that the G-model is designed for the purpose of a fitting data to a model, rather than for the evaluation purposes. Therefore, we design the G-model as an abstract model that maximally simplifies reality by capturing only the essence of the clustered partitioned connectivity. And, the H-model is the one that is designed for the evaluation purposes. Therefore, we design the H-model with much more realistic connectivity assumptions than the G-model. Beside this, the H-model is a parsimonious model with a small number of parameters and thus suitable for the evaluations. Nevertheless, we show evaluation results of IH for both of these models.

The operation of IH relies on the knowledge about the stable CP topology, i.e., the knowledge about CPs and the flows of mobile nodes among these CPs. In many situations, it would be unrealistic to assume that these are known a-priori. Therefore, we develop methods collectively called Collaborative Graph Discovery (COGRAD) that allow the nodes to discover the CP topology without any explicit signals from the environment (such as GPS coordinates or fixed beacons). To our best knowledge we are not aware that any similar problem is considered in the research literature.

We also consider the COGRAD problem under the G-model first. The simplified connectivity assumptions make the problem easier. Though the designed COGRAD scheme does not have much of a practical value, it was helpful in understanding the essence of the problem. Then, we consider COGRAD under the H-model with the relaxed connectivity assumptions. Moreover, we make an effort to design an adaptive COGRAD algorithm for the H-model, that estimates its parameters. The parameters change over time adapting to the current mobility conditions and the current CP topology. Hence, the designed scheme has a practical value beyond the H-model. We show through simulations that the designed COGRAD schemes can replace an oracle with knowledge of the CP topology, allowing IH to operate even in scenarios without any cues from the environment.

Note that in both COGRAD schemes (for the G-model and the H-model), it is the G-model that we use as a model to infer the CP topology. This was exactly the purpose of the designing the G-model. Therefore, in both of these schemes, a CP is associated with a connected component, i.e., no more than one CP can exist in a component. These component-based algorithms have limitations in a mobility scenario when two or more CPs are too close to each other. This is an important scenario that we expect to occur in reality in many circumstances. To overcome these limitations, we propose a soft' labeling approach. This soft approach allows the coexistence of more than one CPs in a component. We show that the soft

labeling algorithm is a more robust algorithm operating successfully under wider range of the model's parameters than the component based one.

One of the important step in evaluations of our IH and COGRAD algorithms is to use a realistic mobility data-set instead of a syntectic mobility model. In this thesis, we did not achieve this step. The reason for this is that there were not suitable data-sets available. This is because we need a large-scale data-set with relatively frequent updates of nodes locations in order to successfully run the COGRAD algorithm. The only data-set that we are aware of which fulfils this requirement is the San-Francisco data-set. We ran the offline and decentralized version of the labeling algorithm under this data-set. But, there were only small number of stable CPs found in the data-set, so that the IH approach was not much beneficial. We argue that the reason for such small number of CPs is that the mobility pattern of taxis is very specific and may not give sufficient evidence of other CPs (located for example nearby sport centers, gas stations, movie theaters etc). Thus we believe that given a large set of mobile traces for the same area, which contains GPS location updates of different types of vehicles, it should provide us with more CPs. We also believe that CPs might be much easier observed at a larger scale, e.g. not at the San Francisco agglomeration scale only, but at the whole Bay area scale. Therefore, to achieve this further step more suitable data-sets are desirable.

One of interesting questions that remained unanswered in this thesis is the applicability and robustness of the soft labeling algorithm. An interesting question is for example to characterize mobility conditions where the soft labeling achieves good results and what are its limitations. Moreover, note that the incorporation of the soft labeling to IH is not straightforward. The possibility of having several CPs in a component requires different handling of routing in IH within a component. This opens further interesting research questions.

In this thesis, we show that both the predictability and the stable and heterogenous node density are important features of realistic mobility patterns for the design of routing algorithms in mobile ad-hoc networks. By exploiting these, we show that it is possible to design efficient routing algorithms for mobile partitioned networks. The possible scenarios of the mobility with CPs other than the observed taxi traces include:

- People in urban environments: workplace, restaurants, public transportation (train stations, airports), movie theaters, etc.
- Wildlife monitoring: watering holes, clearings, oases, etc.
- Office buildings: cafeterias, conference rooms, water coolers, hallways, etc.

Therefore, applicability of our routing methods in the real world is very promising.

Curriculum Vitæ

Natasa Sarafijanovic-Djukic was born in Cacak, Serbia. She received the BSc degree in electrical engineering in 2000 from the Faculty of Electrical Engineering, Belgrade, Yugoslavia. From 2000-2002 she worked as a GSM support engineer at Ericsson d.o.o in Belgrade. From 2002-2003 she attended the Doctoral School in Communication Systems at EPFL, Switzerland. In 2003 she joined the Laboratory for Computer Communications and Applications (LCA) at the Department of Communication Systems, EPFL. From 2003-2009, she worked as a research assistant at LCA towards her PhD on the topic of routing in mobile ad-hoc networks.

Publications

- N. Sarafijanovic-Djukic and M. Grossglauser, "Last Encounter Routing under Random Waypoint Mobility", NETWORKING 04, Athens, Greece, May 2004.
- N. Sarafijanovic-Djukic, M. Piorkowski, and M. Grossglauser, "Island Hopping: Efficient Mobility-Assisted Forwarding in Partitioned Networks", IEEE SECON 06, Reston, VA, September 2006.
- N. Sarafijanovic-Djukic, M. Piorkowski, and M. Grossglauser, "Island Hopping: Efficient Mobility-Assisted Forwarding in Partitioned Networks", tech. rep., 2007.
- M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "On Clustering Phenomenon in Mobile Partitioned Networks", The First ACM SIGMOBILE International Workshop on Mobility Models for Networking Research, Hong Kong, May 2008.
- M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "A Parsimonious Model of Mobile Partitioned Networks with Clustering", COMSNETS 09, Bangalore, India, January 2009.

Bibliography

[AS03] Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Computing*, 7(5):275–286, 2003.

[BBL05] Brendan Burns, Oliver Brock, and Brian N. Levine. MV Routing and Cpacity Building in Disruption Tolerant Networks. volume 1, pages 398–408, 2005.

[BC03] Magdalena Balazinska and Paul Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 303–316, New York, NY, USA, 2003. ACM.

[BG05] Irad Ben-Gal. *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. Kluwer Academic Publishers, 2005.

[BK07] Michel Barbeau and Evangelos Kranakis. *Principles of Ad-hoc Networking*. Wiley Publishing, 2007.

[BLBG05] Ljubica Blazevic, Jean-Yves Le Boudec, and Silvia Giordano. A location-based routing method for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 4(2):97–110, 2005.

[BLV07] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. DTN Routing as a Resource Allocation roblem. *SIGCOMM Computer Communication Review*, 37(4):373–384, 2007.

[Bou] Jean-Yves Le Boudec. Performance Evaluation Lecture Notes.

[BRS05] Sven Bittner, Wolf-Ulrich Raffel, and Manuel Scholz. The area graph-based mobility model and its impact on data dissemination. In *PERCOMW '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 268–272, Washington, DC, USA, 2005. IEEE Computer Society.

[BSH03] F. Bai, Narayanan Sadagopan, and A. Helmy. Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, volume 2, pages 825–835, 2003.

[CBD02] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications & Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[CDB+07] Praphul Chandra, Daniel M. Dobkin, Alan Bensky, Ron Olexa, David Lide, and Farid Dowla. *Wireless Networking: Newnes Know It All*. Newnes, Newton, MA, USA, 2007.

[CHD+07] Augustin Chaintreau, Pan Hui, 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. Fellow-Crowcroft, Jon.

[CJ03] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr), 2003.

[CLL04] Yuanzhu Peter Chen, Arthur L. Liestman, and Jiangchuan Liu. *Clustering Algorithms for Ad Hoc Wireless Networks*. Nova Science Publishers, 2004.

[CM01] Xiangchuan Chen and Amy L. Murphy. Enabling Disconnected Transitive Communication in Mobile Ad Hoc Networks. In *Workshop on Principles of Mobile Computing, colocated with PODC'01*, pages 21–27, Newport, Rhode Island, USA, 2001.

[CR09] Suchismita Chinara and Santanu Kumar Rath. A survey on one-hop clustering algorithms in mobile ad hoc networks. *J. Netw. Syst. Manage.*, 17(1-2):183–207, 2009.

[Dav00] Vanessa Ann Davies. Evaluating mobility models within an ad hoc network, 2000.

[DFGV03a] Henri Dubois-Ferriere, Matthias Grossglauser, and Martin Vetterli. Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks using Encounter Ages. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, New York, NY, USA, 2003. ACM.

[DFGV03b] Henri Dubois-Ferriere, Matthias Grossglauser, and Martin Vetterli. Space-Time Routing in Ad Hoc Networks. In *Proc. 2nd International*

Conference on AD-HOC Networks and Wireless, Montreal, Canada, October 2003.

[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, Washington, DC, USA, 2001. IEEE Computer Society.

[Fal03] Kevin Fall. A Delay-tolerant Network Architecture for Challenged Internets. In *SIGCOMM '03: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 27–34, New York, NY, USA, 2003. ACM.

[FG04] L. Farnell and W. G. Gibson. Monte Carlo Simulation of Diffusion in a Spatially Nonhomogeneous Medium: Correction to the Gaussian Steplength. *Journal of Computational Physics*, 198(1):65–79, 2004.

[FH04] Bai Fan and Ahmed Helmy. *Wireless Ad Hoc and Sensor Networks*, chapter A Survey of Mobility Models in Wireless Adhoc Networks, pages 1–29. Kluwer Academic Publishers, 2004.

[Gar04] Crispin W. Gardiner. *Handbook of stochastic methods : for physics, chemistry and the natural sciences*. Springer, 2004.

[GD04] Robert M. Gray and Lee D. Davisson. *Introduction to Statistical Signal Processing*. Cambridge University Press, December 2004.

[GS07] Venkata C. Giruka and Mukesh Singhal. A self-healing on-demand geographic path routing protocol for mobile ad-hoc networks. *Ad Hoc Networks*, 5(7):1113–1128, 2007.

[GV03] Matthias Grossglauser and Martin Vetterli. Locating Nodes with EASE: Last Encounter Routing in Ad Hoc Networks through Mobility Diffusion. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, San Francisco, CA, USA, April 2003.

[GV06] Matthias Grossglauser and Martin Vetterli. Locating Mobile Nodes with EASE: Learning Efficient Routes from Encounter Histories Alone. *IEEE/ACM Transactions on Networking*, 14(3):457–469, 2006.

[Haa97] Zygmunt Haas. A New Routing Protocol For The Reconfigurable Wireless Networks. In *IEEE 6th International Conference on Universal Personal Communications*, pages 562–566, San Diego, CA,USA, October 1997.

[HABR05] Khaled A. Harras, Kevin C. Almeroth, and Elizabeth M. Belding-Royer. Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks. *IFIP Networking. Lecture Notes in Computer Science*, 3462:1180–1192, May 2005.

[HCY08] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble Rap: Social-based Forwarding in Delay Tolerant Networks. In *MobiHoc '08: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 241–250, New York, NY, USA, 2008. ACM.

[HGPC99] Xiaoyan Hong, Mario Gerla, Guangyu Pei, and Ching C. Chiang. A group mobility model for ad hoc wireless networks. In *MSWiM '99: Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, New York, NY, USA, 1999. ACM.

[HMS+05] Wei-jen Hsu, Kashyap Merchant, Haw-wei Shu, Chih-hsin Hsu, and Ahmed Helmy. Weighted waypoint mobility model and its impact on ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1):59–63, 2005.

[Hsi01] Pai-Hsiang Hsiao. Geographical region summary service for geographical routing. *SIGMOBILE Mobile Computing and Communications Review*, 5(4):25–39, 2001.

[HSPH07] Wei J. Hsu, Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Ahmed Helmy. Modeling time-variant user mobility in wireless mobile networks. In *Proceedings of the 26th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Anchorage, Alaska, March 2007. IEEE.

[JBRAS03] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri. Towards realistic mobility models for mobile ad hoc networks. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 217–229, New York, NY, USA, 2003. ACM.

[JFP04] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a Delay Tolerant Network. In *SIGCOMM '04: Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 145–158, New York, NY, USA, 2004.

[JLS07] Evan P. C. Jones, Lily Li, and Jakub K. Schmidtke. Practical Routing in Delay-Tolerant Networks. *IEEE Transactions on Mobile Computing*, 6(8):943–959, 2007.

[JM96a] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[JM96b] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[JT87] J. Jubin and J. D. Tornow. The DARPA Packet Radio Network Protocols. *IEEE Proceedings*, 75:21–32, January 1987.

[KE05] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. *Wirel. Netw.*, 11(1-2):115–133, 2005.

[KK00] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 243–254, New York, NY, USA, 2000. ACM Press.

[Kle75] Leonard Kleinrock. *Queueing Systems*, volume 1. Jhon Wiley & Sons, New York, 1975.

[KM05] Demetres Kouvatsos and Is-Haka Mkwawa. Broadcasting Methods in Mobile Ad Hoc Networks: An Overview, Tutorial Paper. In *HET-NETs '05: 3rd International Conference on Performance Modeling and Evaluation of Heterogeneous Networks*, Ilkley, West Yorkshire, U.K., July 2005.

[KSH07] Jin-Il Kim, Jeong-Young Song, and Yoon-Cheol Hwang. Location-based routing algorithm using clustering in the manet. In *FGCN '07: Proceedings of the Future Generation Communication and Networking*, pages 527–531, Washington, DC, USA, 2007. IEEE Computer Society.

[KT81] Samuel Karlin and Howard M. Taylor. *A second course in stochastic processes*. Academic Press, 1981.

[KWSB04] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting Places from Traces of Locations. In *WMASH '04: Proceedings of the 2nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, pages 110–118, New York, NY, USA, 2004. ACM.

[lap00] One Laptop Per Child, 2000.

[LDS03a] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[LDS03b] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic Routing in Intermittently Connected Networks. *SIGMOBILE Mobile Computing and Communications Review*, 7(3):19–20, 2003.

[LFC07] Jérémie Leguay, Timur Friedman, and Vania Conan. Evaluating MobySpace-based routing strategies in delay-tolerant networks: Research Articles. *Wireless Communications and Mobile Computing*, 7(10):1171–1182, 2007.

[LJDC+00] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 120–130, New York, NY, USA, 2000. ACM.

[LWM06] Vincent Lenders, Jörg Wagner, and Martin May. Analyzing the Impact of Mobility in Ad Hoc Networks. In *REALMAN '06: Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality*, pages 39–46, New York, NY, USA, 2006. ACM.

[LYD06] Sunho Lim, Chansu Yu, and C.R. Das. Clustered Mobility Model for Scale-Free Wireless Networks. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pages 231–238, Tampa, Florida, USA, Nov. 2006.

[MHM05] Mirco Musolesi, Stephen Hailes, and Cecilia Mascolo. Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks. In *WOWMOM '05: Proceedings of the Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*, pages 183–189, Washington, DC, USA, 2005. IEEE Computer Society.

[MM07] Mirco Musolesi and Cecilia Mascolo. Designing mobility models based on social network theory. *SIGMOBILE Mob. Comput. Commun. Rev.*, 11(3):59–70, 2007.

[MM09] Mirco Musolesi and Cecilia Mascolo. Mobility Models for Systems Evaluation. A Survey. In Benoit Garbinato, Hugo Miranda, and Luis Rodrigues, editors, *Middleware for Network Eccentric and Mobile Applications*, pages 43–62. Springer, February 2009.

[MR96] Ronald Meester and Rahul Roy. *Continuum percolation*. Cambridge University Press, 1996.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers. In *SIGCOMM '94: Proceedings of the Conference on*

Communications Architectures, Protocols and Applications, pages 234–244, New York, NY, USA, 1994. ACM.

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad-hoc On-Demand Distance Vector (AODV) Routing, 2003.

[Pio09] Michal Piorkowski. *Mobility-Centric Design of Collaborative Transportation Systems*. PhD thesis, Lausanne, 2009.

[PP96] M. Penrose and A. Pisztora. Large Deviations for Discrete and Continuous Percolation. *Advances in Applied Probability*, 28(1):29?–52, 1996.

[PSDG09] Michal Piorkowski, Natasa Sarafijanovoc-Djukic, and Matthias Grossglauser. A Parsimonious Model of Mobile Partitioned Networks with Clustering. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1–10, 2009.

[RHB+07] Ram Ramanathan, Richard Hansen, Prithwish Basu, Regina Rosales-Hain, and Rajesh Krishnan. Prioritized Epidemic Routing for Opportunistic Networks. In *MobiOpp '07: Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking*, pages 62–66, New York, NY, USA, 2007. ACM.

[Ris89] H. Risken. *The Fokker-Planck equation : methods of solution and applications*. Springer, 1989.

[RMvdB06] F. Roijers, M.R.H. Mandjes, and J.L. van den BergA. Analysis of congestion periods of an m/m/inf-queue, report pna-e0606, issn 1386-3711. Technical report, Centrum voor Wiskunde en Informatica (CWI), Netherlands, March 2006.

[SB04] David Schwab and Rick Bunt. Characterising the Use of a Campus Wireless Network. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 862–870, Hong Kong, China, March 2004. IEEE.

[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*, pages 41–50, Washington, DC, USA, 2004. IEEE Computer Society.

[SM01] Miguel Sánchez and Pietro Manzoni. Anejos: a java based simulator for ad hoc networks. *Future Gener. Comput. Syst.*, 17(5):573–583, 2001.

[SPR05] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and Wait: an Efficient Routing Scheme for Intermittently Connected Mobile Networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pages 252–259, New York, NY, USA, 2005. ACM.

[SPR08] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Efficient Routing in Intermittently Connected Mobile Networks: the Multiple-copy Case. *IEEE/ACM Transactions on Networking*, 16(1):77–90, 2008.

[SW87] N. Schacham and J. Westcott. Future Directions in Packet Radio Architectures and Protocols. *IEEE Proceedings*, 75:83–99, January 1987.

[TB00] Diane Tang and Mary Baker. Analysis of a local-area wireless network. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 1–10, New York, NY, USA, 2000. ACM.

[TB02] Diane Tang and Mary Baker. Analysis of a metropolitan-area wireless network. *Wirel. Netw.*, 8(2/3):107–120, 2002.

[ter00] TerraNet AB, 2000.

[THB+02] Jing Tian, Joerg Haehner, Christian Becker, Illya Stepanov, and Kurt Rothermel. Graph-based mobility model for mobile ad hoc network simulation. *Simulation Symposium, Annual*, 0:0337, 2002.

[The03] S. Theodoridis. *Pattern Recognition*. Elsevier, 2003.

[TML08] Y. Toor, P. Muhlethaler, and A. Laouiti. Vehicle Ad Hoc Networks: Applications and Related Technical Issues. *Communications Surveys & Tutorials, IEEE*, 10(3):74–88, 2008.

[TZZ03] Kun Tan, Qian Zhang, and Wenwu Zhu. Shortest Path Routing in Partially Connected Ad Hoc Networks. volume 2, pages 1038–1042 Vol.2, 2003.

[VB00] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical report, Duke University, 2000.

[WCYZ07] Yi Wang, Hairong Chen, Xinyu Yang, and Deyun Zhang. Cluster based location-aware routing protocol for large scale heterogeneous manet. In *IMSCCS '07: Proceedings of the Second International Multi-Symposiums on Computer and Computational Sciences*, pages 366–373, Washington, DC, USA, 2007. IEEE Computer Society.

[WHAB09] Mike P. Wittie, Khaled A. Harras, Kevin C. Almeroth, and Elizabeth M. Belding. On the implications of routing metric staleness in delay tolerant networks. *Computer Communications*, 32(16):1699–1709, 2009.

[WJMF05] Yong Wang, Sushant Jain, Margaret Martonosi, and Kevin Fall. Erasure-coding based Routing for Opportunistic Networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pages 229–236, New York, NY, USA, 2005. ACM.

[WLB05] Jörg Widmer and Jean-Yves Le Boudec. Network Coding for Efficient Communication in Extreme Networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pages 284–291, New York, NY, USA, 2005. ACM.

[YLN03] J. Yoon, M. Liu, and B. Noble. Random Waypoint Considered Harmful. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1312–1321, April 2003.