LEVERAGING CLUSTERING FOR EFFICIENT COMMUNICATION IN OPPORTUNISTIC NETWORKS

BY

MEHEDI BAKHT

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Abstract

The overarching goal of my research is to design protocols for efficient communication in delay tolerant networks (DTNs), with a particular focus on pocket switched networks (PSNs), an emerging class of ad hoc networks made up of smartphones and other portable mobile devices carried by humans. Communication in such an infrastructure-less scenario is inherently opportunistic since it relies extensively on detecting, as well as utilizing, unplanned encounters between nodes. While existing solutions look at the nodes in isolation, we propose that the clustering of nodes, which is a common phenomenon observed in different types of DTNs including PSNs, can be leveraged for significantly improving the efficiency of different components of opportunistic communication.

The first step in enabling opportunistic communication is neighbor discovery. In this context, we have developed Searchlight, an asynchronous neighbor discovery protocol that uses systematic probing to considerably decrease discovery latency while allowing nodes to operate at low duty cycles.

However, for an individual node, performing continuous neighbor discovery can still be too expensive with a high-power radio like 802.11. On the other hand, relying only on a low-power, short-range radio for detecting neighbors will result in significantly fewer available contacts. To mitigate this problem, we have developed a scheme for more efficient neighbor discovery that leverages the clustering of nodes as well as the radio heterogeneity of mobile devices. The basic idea is that coordination over a low-power, short-range radio can help clustered nodes distribute the load of scanning over the high-power, long-range radio. We have implemented the protocol successfully on a testbed of Android phones.

Clustering can be also leveraged at a higher level for efficient forwarding of messages. Most routing protocols for DTNs only focus on one-hop encounters. However clustering creates islands of connectivity where path-based routing can be more efficient. Based on this insight, we have developed a lightweight clustering-based routing protocol that performs well in both partitioned and clustered environments by integrating store-carry-forward techniques with path-based techniques when appropriate.

Dedicated to my late grandfathers Dr. Jamshed Bakht and Ajmal Hossain.

Acknowledgments

First and foremost, I want to thank my advisor Robin Kravets. Along with her guidance and advising, it was her faith in me that kept me going during tough times.

I would like to thank my committee members: Tarek Abdelzaher, Klara Nahrstedt and Mostafa Ammar for their time, interest, and helpful comments.

I am very grateful for the guidance and encouragement I have received from Al Harris III, Cidgem Sengul, Rumi Shammin, Shamsi Tamara Iqbal, Munawar Hafiz, Mohammad Maifi Khan, Ragib Hasan, and Fariba Khan. I would like to especially thank my research groupmates Nat Thompson, Samuel Nelson, Farhana Ashraf, Riccardo Crepaldi, and Matt Trower - all of whom have helped me immensely in different stages of my research. Thanks to my fellow graduate students for their support all along: Shameem Ahmed, Yusuf Sarwar, Ahmed Khurshid, Imranul Haque, Sonia Jahid, Abdullah Nayeem, and Ahsan Arefin.

I am very grateful to the Champaign-Urbana Bangladeshi community for the support I have received over the last six years. Thanks to my “family” in Champaign-Urbana for making me and my wife feel at home away from home: Mohammad Sharif Ullah, Nadia Nasreen, Naajid Sharif, Syeda Akter, Khandaker Tareque Islam, Tabeeb, Tashriq, Taher Saif, Shahneela Chowdhury, Farzad, and Faiza.

Thanks to my parents Dr. Zaid Bakht and Halima Hossain, my in-laws, my sister Meherin Reza, my brothers-in-law Raihan Huq and Asheque Al Mahbub, my sister-in-law Laila Tabassum, my niece Nusayeeba Zohra, my grandmothers Zebunessa Khatun and late Sajeda Hossain, my aunts Taslima Hossain, Farzana Hossain, Leann Bakht, and Corazon Bakth, my uncles Nazmul Hossain, Amjad Hossain, Shamsher Bakth, Shoaib Bakht, and Aziz Sultan, my cousins, and other friends and family members who helped me with their support and prayers from afar.

Last, but by no means the least, I am thankful to my wonderful wife Karishma who showed great patience over the last six and a half years to stick by with me.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mobile devices capable of communicating wirelessly are everywhere. This includes smartphones [20] or mobile sensors [29] carried by humans as well as vehicles equipped with Wi-Fi radios [17]. The ubiquity of such devices has created huge potential for networks based on ad hoc connectivity without assuming any end-to-end path between any pair of nodes. These networks, generally known as disruption tolerant networks (DTNs), encompass a large range of environments including, but not limited to, emergency response networks [41], pocket-switched social networks [20], and vehicular networks [17]. A defining characteristic of most of these networks is that communication is completely dependent on exploiting opportunities arising from unplanned and unpredictable encounters between nodes. To distinguish from DTNs that assume controlled mobility (e.g., message ferries [69, 70]) or scheduled connectivity (e.g., inter-planetary communication [19]), we call these networks *opportunistic networks*.

While protocols for opportunistic networks have generally focused on supporting routing messages from one node to another, there is also a growing trend to leverage opportunistic communication for finding "friends" [52], to support mobile social networking [34] or enable content dissemination [66]. Recent surveys have showed that there is a strong demand for co-located people to communicate and share information about the surroundings with each other via personal mobile devices [36, 68]. Feedback from these surveys also suggests that people may even share personal content or find content that the public media cannot offer. For example, people on a subway may like to read others' blogs [36], since the infrastructure is often unavailable.

The success of such opportunistic communication depends on a two-step process. The first step is to ensure that any contact opportunity between nodes get detected successfully. Given that most of the nodes are energy-constrained and lack synchronization, energy-efficient neighbor discovery with low latency is challenging, especially for nodes with high-power radios like Wi-Fi. Once a contact gets detected, the next step is to ensure that the right scheme is used to exchange messages during that contact opportunity, so that there is a high probability of a message reaching its intended destination through such chance encounters without draining resources like storage and energy.

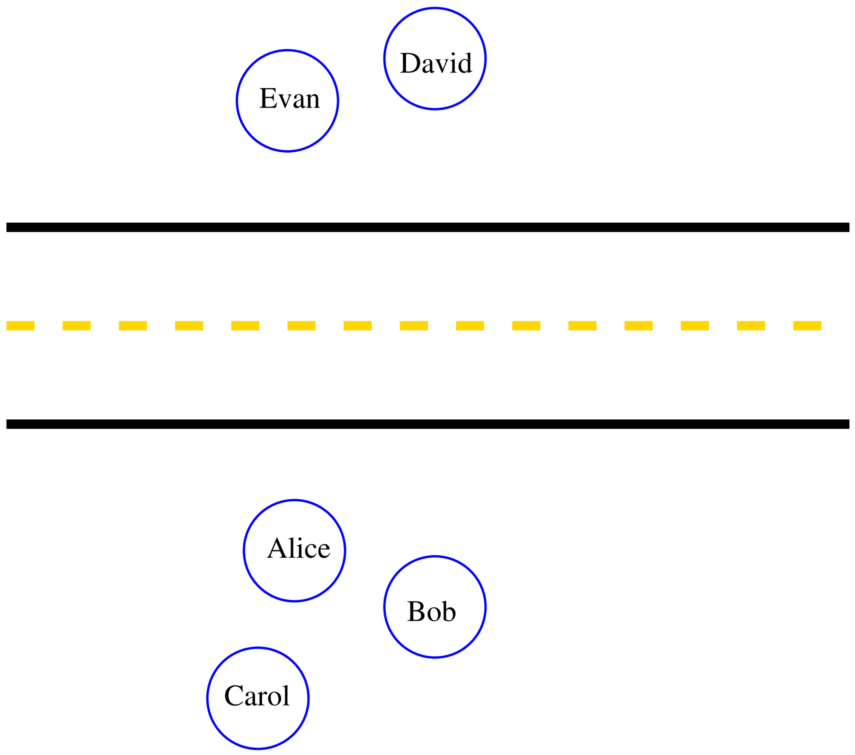


Figure 1.1: Opportunistic Networking

We next present two examples to motivate the importance of the above mentioned two steps. First, consider three customers Alice, Bob and Carol in a cafe, each of whom is carrying a mobile device with two types of radios - Bluetooth and Wi-Fi (see Figure 1.1). Since they are in close vicinity of each other, there is an opportunity for ad hoc communication over the Bluetooth radio, whose range is around 10 meters. Because of the comparably low energy cost of Bluetooth scanning, we assume that these contact opportunities get discovered without expending too much energy. However, Alice may actually be looking for David (probably because she has a message for him or he has a content that she is interested in), who is sitting with Evan at the fast food shop across the street. The distance between Alice and David is more than the Bluetooth range, but well within the Wi-Fi transmission range of 100 meters. However, to save energy, neither Alice nor David uses the Wi-Fi radio of their mobile devices to search for neighbors, and the contact opportunity goes to waste.

For the second example, assume that Alice somehow discovers Evan through her Wi-Fi radio and they remain connected for a short duration. Since current opportunistic protocols only focus on utilizing one-hop connections, Alice does not learn that her intended destination David is just one hop away from Evan and she could have delivered the message to Evan to forward it to David.

Our key observation is that in both scenarios, more efficient communication would have been possible if the nodes, instead of acting completely on their own, had cooperated with nearby peers. Such cooperation is often possible since nodes have a natural tendency to cluster in a variety of environments. Protocols for opportunistic networks should be aware of and work to take advantage of this phenomenon, instead of being oblivious to it. We propose leveraging this phenomenon to allow *efficient* detection of contact opportunities, as well as *effective* utilization of these opportunities for routing messages with high-delivery ratios and low end-to-end delay, while incurring low resource overhead.

1.1 Scope of Opportunistic Networking

Opportunistic networking enables communication in a variety of infrastructure-less scenarios, among which the area of Pocket Switched Networks (PSN) [26] has garnered most attention in recent years. Since this communication paradigm exploits contact opportunities between mobile devices and human mobility to transfer data in a peer-to-peer fashion, it has created a huge potential for a whole range of mobile social networking applications. A common feature across all of these applications is the need to detect "nearby" peers [1, 3, 52]. Once detected, the opportunity to contact other co-located nodes can be exploited for a variety of purposes that range from turning a user's device into a localized content server (e.g., Lokast [4]) to more social network-oriented goals like finding people with similar interests and hobbies [2]. While energy-efficient discovery is particularly critical for networks composed of battery-powered devices, routing messages to intended destinations using the limited contact opportunities is a challenge common to *any* opportunistic network. For example, in an emergency response scenario where the primary infrastructure, such as centralized cellular services, is either down (e.g., destroyed) or unusable (e.g., overloaded) [42], opportunistic message passing may be the only viable means for establishing communication. Thus, any comprehensive effort for improving opportunistic communication must focus on supporting both contact detection as well as efficient unicast routing of messages.

1.2 Necessary Components and Research Contributions

The goal of this work is to enable efficient communication in infrastructure-less scenarios by focusing on the two essential components of opportunistic communication: (1) detection of contact opportunities and (2) routing. Clustering can be leveraged to improve the performance of both these components. For detecting encounters, the first step is to perform energy-efficient asynchronous neighbor discovery over a single radio. More contact opportunities can be discovered if a cluster of dual-radio devices can cooperate over the low-power radio to find neighbors that are reachable only through the high-power radio.

Successful discovery leads to the second step of opportunistic communication where messages are exchanged and forwarded to enable DTN routing. Again, creation of multi-hop partial paths within clusters can be leveraged to augment existing DTN routing protocols with cluster-aware mechanisms for more efficient utilization of contact opportunities. While both discovery and routing for opportunistic networks has been studied in literature, they have been mostly done in a cluster-agnostic way, rendering them comparably inefficient to a cluster-aware approach, which is the focus of our research.

1.2.1 Understanding the Environment: Clustering in Opportunistic Networks

Research on opportunistic networks has largely focused on scenarios where nodes remain completely isolated from each other most of the time, except for occasional single-hop encounters. However, most opportunistic networks do not lie at such an extreme end of the connectivity continuum. Instead, nodes are heterogeneously distributed in space, resulting in the formation of small islands of connectivity, or *clusters*, although the overall network may remain partitioned. Recent studies of real-life traces of vehicular mobility, as well as human contacts, have confirmed the existence of such dynamic clustering in different environments. Discovery of this phenomenon has warranted a rethinking of our approach to opportunistic communication. The fact that nodes are often in the vicinity of other nodes can be leveraged to improve both the discovery and the routing process.

1.2.2 Energy-efficient Asynchronous Neighbor Discovery

The success of applications designed for opportunistic networks lies in the ability of a node to efficiently discover the presence of other nodes in its transmission range. Although discovery is also challenging in environments like ad hoc and sensor networks, there is some expectation of stability in the network and so applications expect connectivity and deal with failure as it happens. There is also some expectation of sufficient density so that when a node has data to send, some number of neighbors will be available at the time of discovery. However, opportunistic networks do not have the same expectation of stability or node density. Instead, nodes look for a neighbor and then decide to communicate or send data. This contact-driven communication puts a heavy burden on the discovery protocols, especially in terms of energy. To meet this challenge, we have designed an asynchronous neighbor discovery protocol that strikes a balance between low-power operation and small discovery latency. We further increase the performance of the protocol through the novel introduction of asymmetric slots that brings down the cost of discovery to almost half in comparison to the nearest performing protocol.

1.2.3 Finding more Neighbors through Cooperation

For contact-driven applications, the more neighbors that can be discovered, the better the performance of these applications will be. However, for a device running on battery, and hence a limited energy budget, searching for neighbors through a high-power radio like Wi-Fi becomes very expensive energy-wise, even with duty-cycling. Therefore, current approaches designed for multi-radio devices dismiss the use of Wi-Fi for discovery completely [52] and rely only on a low-power low-range radio like Bluetooth to detect encounters, which results in far fewer contact opportunities. To address this problem, we propose leveraging clustering. Given the typical clustering in social settings, our scheme enables nodes equipped with multiple radios to coordinate with nearby nodes through the low-power radio and share the cost of discovery through the high-power long-range radio. This ensures that neighbors reachable only through the high-power radio get detected at acceptable energy cost.

1.2.4 Clustering-aware Adaptive Routing

After successfully detecting an encounter, a node uses one of the many existing DTN protocols to decide which messages it should forward during that contact opportunity. However, most of these protocols focus only on single-hop connectivity. In many opportunistic networks, nodes are not homogeneously distributed in space and so tend to assemble into clusters of varying size and longevity. This clustering often presents multi-hop routing opportunities that the current opportunistic protocols are incapable of taking advantage of. To mitigate this deficiency, we propose augmenting existing DTN routing protocols with a clustering-aware component that intelligently integrates store-carry-forwarding and path-based routing on top of a lightweight clustering substrate.

1.3 Proposal Structure

In Chapter 2, we analyze the clustering phenomenon in opportunistic networks in more details. Chapter 3 presents a new energy-efficient asynchronous protocol for neighbor discovery. Chapter 4 presents our coordinated discovery protocol that enables multi-radio devices to search for neighbors through both radios. In Chapter 5, we describe a cluster-aware technique to enhance current unicast protocols, which makes it possible to adaptively use path-based routing or store-carry-forward-based approach as appropriate. Finally, we conclude in Chapter 6 and discuss possible extensions to the work based on integration of discovery and message forwarding.

Chapter 2

The Clustering Phenomenon

Clustering in opportunistic networks has gained lot of attention lately. Evidence of cluster formation has been found by looking at instantaneous snapshots of connectivity graphs from different traces [53]. It was shown that the spatial distribution of nodes in an urban environment is heavy tailed, which implies that some places have a population density much above the average. Another recent work [25] used Monte Carlo simulations to show that even for disconnected networks composed of homogeneously distributed nodes, there is a significant region of network density where a substantial portion of nodes are connected by multi-hop paths. In real-life networks, the spatial distribution of nodes is less uniform and hence creates more clusters. Examples include campus scenarios where nodes accumulate in areas of interest (e.g., library, cafeteria, classrooms) with less connectivity available between these areas [27]. Vehicular networks also exhibit similar tendency of nodes to gather at specific locations (e.g., due to decelerating or stopping at junctions or traffic lights [56]).

Given the existence of clustering in opportunistic networks, the next step is to better understand the clustering process by looking at how the connectivity graph evolves *over time.* This is challenging in a highly mobile environment since it requires establishing correspondence between connected components from different time instants. The Centralized Cluster Labeling (CCL) algorithm [56] addresses this problem by assigning labels for connected components intelligently with the goal of ensuring that sequences of corresponding components over time get the same label. Using this algorithm, two metrics, cluster lifetime and cluster size, were introduced to analyze the cluster dynamics over time [55]. Cluster lifetime denotes the time a cluster is present in the network and cluster size denotes the average number of nodes of a cluster over its lifetime. The mobility traces of taxi cabs in San Francisco, which contained GPS coordinates of approximately 500 taxis collected over 30 days in the San Francisco Bay Area, was analyzed for these two metrics. It was found that the cluster lifetime for the trace spanned from 10 seconds up to one day and was heavy tailed. For the second metric, cluster size, the values varied from 2 to as high as 200. However, the two metrics were not studied in tandem. For example, the analysis tells us that at least one cluster had an average of around 200 nodes during its lifetime, but does not tell us how long that particular cluster lasted. This is important because larger clusters facilitate more path-based routing, but only if they are reasonably stable.

A more comprehensive study of cluster dynamics was presented in [54], which was also based on the same San Francisco cab trace. Particular focus was given to clusters that stay alive for longer periods than the average lifetime and that their sizes exceed the average size of a cluster. By analyzing the cluster size evolution of the nine longest lasting clusters, each of which had at least 20 vehicles on average, the authors were able to identify clusters that survived even up to 120 minutes. However, checking how long a cab remained in a cluster yielded interesting results. For six of those nine clusters, the median association time of a node to a cluster was found to be close to 1% of a cluster's lifetime, while for the remaining three it was between 5 to 10%. This suggests that clusters that get formed are stable in size and duration, but not necessarily in member identities.

2.1 Impact of Multi-hop Connectivity on Contact rate

The performance of opportunistic protocols usually increases when nodes get more contact time with each other. Current approaches consider only single-hop direct connections as contacts. We present a simulation based analysis for evaluating how taking into consideration multi-hop connectivity impacts the "contact" time between nodes.

2.1.1 Simulation Settings

To understand clustering better, we looked at structure-less networks with random mobility, since random networks would make an interesting baseline for cluster-based solutions. We evaluated the behavior of a random network in the ONE [31] simulator where 250 nodes followed the Random Waypoint movement on a $5600m \times 4000m$ rectangular area for an hour. The speeds of the nodes were chosen randomly between $5m/s$ and $25m/s$ and the wait times were uniformly chosen between 60 and 120 seconds.

2.1.2 Results

We computed the total contact time in two ways. The first method considered single-hop connectivity, where a connection opportunity $C(i,j,t)$ between i and j at time t exists only if there is a direct link (single hop) between i and j at time t. Formally,

$$C(i,j,t) = \begin{cases} 1 \text{ if } link(i,j) \text{ is active at time, } t \\ 0 \text{ otherwise.} \end{cases}$$

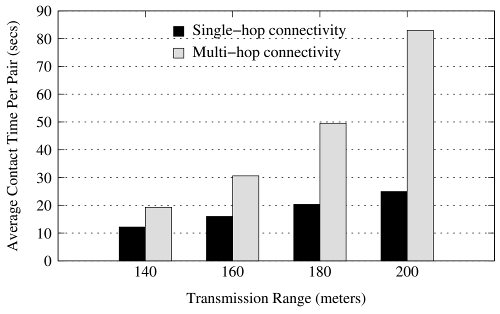


Figure 2.1: Impact of multi-hop connectivity on contact time

For total simulation time T and total number of nodes n, we then define the average per-pair contact time in the following way, $\textit{Avg. per pair contact time} = \dfrac{\sum_i \sum_j \int_0^T C\left(i, j, t\right) dt}{n\left(n-1\right)}$.

The second method addressed multi-hop connectivity, where $C\left(i, j, t\right) = 1$ only if there is a direct path (of one or more hops) between i and j. To find such contacts, we created a time-varying connectivity graph from the mobility trace and ran a depth first search to find all connected components whenever a connection was established or torn down. Figure 2.1 compares the average per-pair contact times observed by both methods for four different transmission ranges. When multi-hop paths are considered, even in completely random networks, the average observed connectivity between each pair of nodes increases by at least 50%, even for small transmission ranges (low density).

2.2 Conclusions and Future Directions

Existing works have provided evidence of clustering in real life vehicular networks. In order to take advantage of this phenomenon, it is essential to know how it impacts reachability between individual pair of nodes. Our simulation-based analysis takes a first step towards understanding this dynamics. The next step would be to perform similar analysis on real-life vehicular traces. Other aspects of clustering, including determination of cluster diameters, effect of radio heterogeneity, and inter-cluster encounters, also need to be investigated to enable us better understand how clustering can be leveraged for more efficient opportunistic communication.

Chapter 3

Efficient Asynchronous Neighbor Discovery: *Searchlight*

The usefulness of applications designed for opportunistic networks is limited by the lack of effective and energy efficient neighbor discovery protocols. While probabilistic approaches perform well for the average case, they exhibit long tails resulting in high upper bounds on neighbor discovery time. Recent deterministic protocols, including Disco and U-Connect, improve on the worst case bound, but do so by sacrificing average case performance. In response to these limitations, we have designed Searchlight, a highly effective asynchronous discovery protocol that is built on three basic ideas. First, it leverages the constant offset between periodic awake slots to design a simple probing-based approach to ensure discovery. Second, it allows awake slots to cover larger sections of time which reduces total awake time drastically. Finally, Searchlight has the option to employ probabilistic techniques with its deterministic approach that can considerably improve its performance in the average case when all nodes have the same duty cycle. We validate Searchlight through analysis and a series of simulation and real-world experiments on smartphones that show considerable improvement (up to 50%) in worst-case discovery latency over existing approaches in almost all cases, irrespective of duty cycle symmetry.

3.1 Energy-efficient Asynchronous Neighbor Discovery Protocols

Asynchronous neighbor discovery algorithms mostly work on a time-slot basis, where time is assumed to be divided into slots of equal size and all nodes agree on the size of a slot. Based on the protocol used, nodes decide to remain awake during specific slots, which are called *active* slots, and sleep during the remaining slots. During an active slot, the node may send/receive or do both, depending on application requirements. Successful discovery takes place between two neighbors whenever their active slots overlap. To be energy efficient, a discovery scheme needs to use as few active slots as possible to discover neighbors within a reasonable time limit. Current approaches to energy-efficient asynchronous neighbor discovery fall broadly into three categories - probabilistic, quorum-based, and deterministic. The relative strengths and weaknesses of these existing schemes can be judged by answering the following three questions:

- Flexibility: Can the protocol handle both symmetric and asymmetric duty cycles?
- Average-case Performance: Does the protocol do well in *most* of the cases?
- Worst-case Latency: Does the protocol provide an acceptable strict bound on the worst-case discovery latency?

Most well known among probabilistic approaches is a family of "birthday protocols" [37] where nodes transmit/receive or sleep with different probabilities. This scheme works well in the average case and allows asymmetric operation. However, the main drawback of the birthday protocol is its failure to provide a bound on the worst case discovery latency, leading to long tails on discovery probabilities.

In the Quorum-based protocols [33, 63], time is divided into sets of m^2 contiguous intervals. These m^2 intervals are arranged as a 2-dimensional $m \times m$ array and each host can pick one row and one column of entries as awake intervals. This ensures that no matter what row and column are chosen, any two nodes have at least two overlapping awake intervals. While the Quorum protocol provides a reasonable bound on worst-case latency, it performs much worse than the probabilistic approach in the majority of the cases. Also, the initial approach [63] lacks flexibility since m is a global parameter and hence supports only symmetric operation. Lai et al. [33] improved that scheme to handle asymmetric cases when there are only two different schedules in the entire network. Another approach that works mainly for the symmetric case is the application of block design using difference sets to the problem of asynchronous neighbor discovery [71]. For the asymmetric case, designing the appropriate schedule following the proposed scheme becomes similar to the vertex-cover problem, which is an NP-complete problem.

Deterministic protocols overcome this limitation of being applicable to only symmetric cases and can handle both symmetric and asymmetric operation while still providing a strict bound on worst-case latency. In Disco, each node chooses a pair of prime numbers such that the sum of their reciprocals is as close as possible to the desired duty cycle. The nodes then wake up at multiples of the individual prime numbers. If one node chooses primes p_1, p_2 and another node chooses p_3, p_4, the worst-case discovery latency between these two nodes will be $min\{(p_1 \cdot p_3), (p_1 \cdot p_4), (p_2 \cdot p_3), (p_2 \cdot p_4)\}$, provided the two primes in the pair are not equal. A more recent deterministic approach, U-Connect [29], uses a single prime per node. Instead of just waking up only 1 slot every p slots, the nodes also wake up $\frac{p+1}{2}$ slots every p^2 slots. The worst-case latency for U-Connect is p^2, which is similar to Disco. However, for the energy-latency product, a metric proposed by the authors to evaluate energy-efficiency of asynchronous neighbor discovery protocols, U-Connect provably fares better than Disco in the symmetric case. Although these deterministic protocols have good worst-case performance, in the majority of cases, they are worse than the birthday protocol.

To successfully meet all of the goals of flexibility, good average-case performance and reasonable worst-case latency, we present a new protocol named Searchlight. Searchlight follows a deterministic approach and provides a strict bound on worst-case latency, which is provably *better* than existing protocols in the symmetric case and similar when duty cycles are asymmetric. Additionally, it also incorporates randomization techniques that result in discovery latency very close to the probabilistic approach in the average case.

3.2 Searchlight

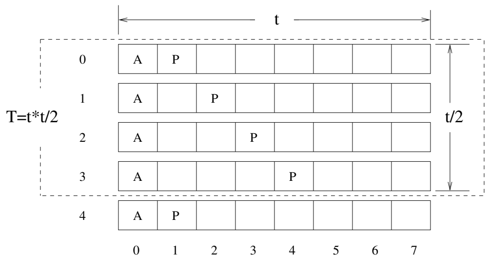


Figure 3.1: Searchlight with sequential probing (t=8)

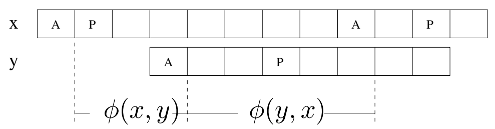


Figure 3.2: Phase offsets (t=8)

Searchlight is a periodic slot-based discovery scheme where a period consists of t contiguous slots. t is known as the period length and is determined based on the specific duty cycle that a node wants to operate at. In every period, there are two active slots - an *anchor* slot (A) and a *probe* slot (P). The position of the anchor slot is fixed and it is the first slot (slot 0) in a period. If t is the same for two nodes, then the anchor slots overlap only if the relative phase offset is less than one timeslot. For all other offsets, the two anchor slots would never meet (assuming no clock drift) since the offset would remain constant during the encounter. This means that the relative position of the anchor slot of one node will remain the same with respect to that of the other node and will be in the range $[1, t-1]$.

The design of Searchlight is based on this key observation about the constant relative offset. Essentially, Searchlight introduces an additional active slot, called a probe slot, in each period. The objective of the probe slot is to search for the anchor slot of the other node in a systematic way. While the constant relative offset only holds in the symmetric case, we show Searchlight also operates in the asymmetric case. We next describe two approaches for determining the schedule for the probe slots.

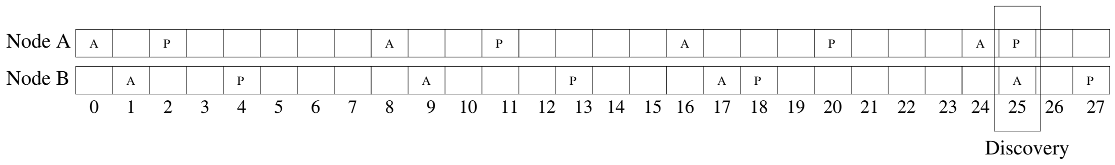


Figure 3.3: Overlap with sequential probing (t=8)

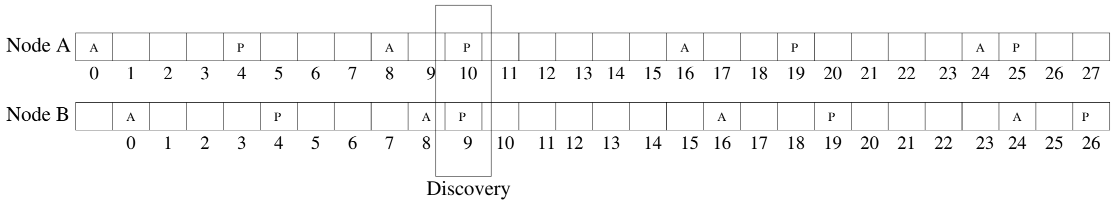


Figure 3.4: Overlap with randomized probing (t=8)

3.2.1 Sequential Probing

In the sequential mode, referred to as Searchlight-S, the position of the probe slot is determined by a counter that starts at 1, increments by 1 every period, ends at $\lfloor \frac{t}{2} \rfloor$ and then starts at 1 again (see Figure 3.1). In other words, if P_i denotes the position of the probe slot in the i-th period, then

$$P_{i+1} = ((P_i + 1)\% \lfloor \frac{t}{2} \rfloor) + 1. \tag{3.1}$$

The position of the probe slot actually follows the pattern $\{1, 2, ..., \lfloor \frac{t}{2} \rfloor\}$ and this pattern gets repeated every $\lfloor \frac{t}{2} \rfloor$ periods, which we call the hyper-period T. For example, for $t = 9$, Searchlight-S uses the pattern {1,2,3,4} to determine the position of the probe slot in each period.

3.2.2 Randomized Probing

In randomized probing, called Searchlight-R, a node systematically moves around its probe slot to find the anchor slots of its neighbors. While this does not affect a node's ability to find an anchor slot, it enables this probe to find other probe slots. Essentially, if the probe slots of two nodes follow the same pattern, they will be in sync with each other, which greatly reduces the probability of a probe-probe overlap. By randomizing the probing, Searchlight benefits from the ideas of the Birthday Protocols and enables probe slots to overlap with a higher probability.

We illustrate this point with the following example (see Figure 3.6). Assume that two nodes A and B are neighbors, they have a relative phase offset of one slot, and $t = 8$. When they first meet, A's probe slot is at position 2, while B's at position 3 (with respect to its own anchor slot). Since both A and B follow the pattern {1,2,3,4}, the next positions of the probe slots will be 3 and 4 respectively. Thus, A and B's probe slot "chase" each another without ever overlapping at any point.

In comparison, Searchlight-R allows nodes to *randomly* pick any permutation of values from 1 to $\lfloor \frac{t}{2} \rfloor$ as the pattern for moving the probe slot. For example, for $t = 9$, instead of being restricted to just {1,2,3,4}, nodes can choose any pattern which is a permutation of the integers 1,2,3 and 4 (e.g., {1,4,3,2},{1,2,4,3}). This modification creates the possibility for two neighbors to pick different patterns, resulting in increased probability of discovery through overlap between the probe slots.

Going back to the earlier example, assume that A randomly chooses the pattern {1,4,2,3} while its neighbor B, chooses the pattern {1,3,2,4} (see Figure 3.7). When they first meet, the probe slots of A and B are at positions 4 and 4 respectively and their relative offset is 1 slot. Because of the phase offset, the probe slots miss each other initially but meet in the next period when A's probe slot moves to position 2 and B's slot moves to position 1. Thus, the use of different patterns instead of the sequential one results in quicker discovery through overlap of probe slots.

3.2.3 Discovery Latency

In this section, we present the analysis for worst-case discovery latency under Searchlight for the symmetric case.

Theorem 3.2.1. *The worst-case discovery latency under Searchlight with parameter t is equal to* $\frac{t^2}{2}$ *slots.*

Proof: For two nodes x and y, let $\phi(x, y)$ be the phase offset (in slots) from the anchor slot of x (A_x) to the anchor slot of y (A_y). Similarly, let $\phi(y, x)$ be the phase offset from A_y to A_x (see Figure 3.2).

Clearly,

$$\phi(x, y) + \phi(y, x) = t. \tag{3.2}$$

In the symmetric case, where both nodes use the same t, $\phi(x, y)$ and $\phi(y, x)$ remain constant during the contact. It follows from Equation (3.2) that $min(\phi(x, y), \phi(y, x)) \leq \frac{t}{2}$.

For both Searchlight-S and Searchlight-R, the probe slot goes through all positions from 1 to $\frac{t}{2}$ every $\frac{t}{2}$ periods. Now, let us denote the probe slots of x and y as P_x and P_y respectively. P_x will meet A_y within $\frac{t}{2}$ periods as long as $1 \leq \phi(x, y) \leq \frac{t}{2}$. Similarly, P_y will meet A_x within $\frac{t}{2}$ periods as long as $1 \leq \phi(y, x) \leq \frac{t}{2}$.

Since at least one from ($\phi(x, y)$,$\phi(y, x)$) is guaranteed to be less than or equal to $\frac{t}{2}$, the worst-case latency can be at most $\frac{t}{2}$ periods or $\frac{t^2}{2}$ slots.

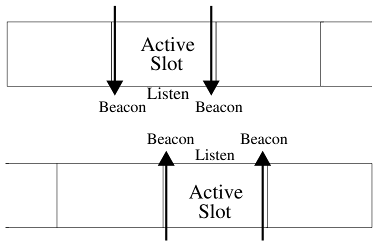


Figure 3.5: Beaconing in an active slot

3.2.4 Slot non-alignment

For illustration purposes, we have shown the slots to be aligned. However, Searchlight is designed as an asynchronous protocol and hence does not assume or rely on the alignment of slot boundaries at different nodes. To ensure that an overlap between two active slots always leads to discovery, Searchlight employs the same beaconing strategy as Disco [22]. A beacon gets sent both at the beginning and end of an active slot and the node remains in listening mode in the intermediate period (see Figure 3.5). Because of this approach, non-alignment of slot boundaries actually results in lower discovery latency in general, since each slot overlaps with two slots of the other node and discovery can happen from either of the two overlaps.

However, two overlaps are redundant when discovery can be ensured with just one overlap. Based on this observation, we next describe striped probing, a strategy that improves the performance of Searchlight significantly by doing more efficient probing.

3.2.5 Striped probing

Probing of every slot from 1 to $\lfloor \frac{t}{2} \rfloor$ guarantees two overlaps every $\lfloor \frac{t}{2} \rfloor$ periods whenever the slot boundaries of two nodes are not aligned. To reduce this redundancy, the probe slot can instead probe every even slot, by using a counter that starts at 2, increments by 2 every period up to $(2 \cdot \lceil \frac{\lfloor \frac{t}{2} \rfloor}{2} \rceil))$ and then starts at 2 again. In other words, if P_i denotes the position of the probe slot in the i-th period, then

$$P_{i+1} = ((P_i) \mod (2 \cdot \lceil \frac{\lfloor \frac{t}{2} \rfloor}{2} \rceil)) + 2. \tag{3.3}$$

However, such striped probing would not work in the rare case when slot boundaries are completely aligned. To handle that case, each active slot "overflows" by δ, a small fraction of the regular slot width. This basically means that if a regular slot size is x, then every active slot (both anchor and probe) become $x(1 + \delta)$ long, and length of the regular slot that follows is reduced to $x(1 - \delta)$. The value of δ should be the smallest possible overlap that guarantees discovery. Because of factors like message transmission time, clock drift, etc., the specific value of *delta* will be largely platform-dependent.

The main advantage of striped probing is that it significantly reduces the worst-case bound. Using the approach detailed in Section 3.2.3, it is easy to show that the worst-case bound for the symmetric case changes to $t \cdot \lceil \frac{\lfloor \frac{t}{2} \rfloor}{2} \rceil$ slots.

3.2.6 Asymmetry

The main design of Searchlight is based on the constant relative offset between the anchor slots of any two neighboring nodes. But this may or not hold in the asymmetric case, when nodes can choose different values of t based on their own energy requirements. One possible approach to guarantee overlap within a fixed bound is to require that t should always be prime. Restricting t to be a prime number, similar to Disco and U-Connect, ensures that for any two nodes operating at different duty cycles, their period lengths t_1 and t_2 will be relatively prime, i.e., they will have no common factors other than 1. Since period lengths are relatively prime, it follows from the Chinese Remainder Theorem [44] that the two anchor slots should overlap at least once every $t_1 \cdot t_2$ slots, where t_1 and t_2 are the two different period lengths.

Another approach is to ensure that even with different values of t, the relative offset between the anchor slots of two nodes remain constant. While this assumption about constant offset holds in the symmetric case for any value of t, keeping relative distance of anchor slots constant in the asymmetric case is much more challenging. This can be done if we can ensure that for any pair of different t values, the larger value is always an integer multiple of the smaller one. Without any loss of generality, let us assume that two nodes A and B use periods t_A and t_B respectively, where $t_A \geq t_B$. If $t_A = n \cdot t_B$, where n is any integer greater than 1, B's anchor slots would always have the same relative distance from A's anchor slot. This ensures that the probe slot of A would eventually meet with an anchor slot of B. This obviously reduces the flexibility in choosing a duty cycle. However, there are two points to consider.

- All other deterministic protocols that can handle both symmetric and asymmetric cases require their protocol parameters to be prime, for *both* symmetric and asymmetric cases. This requirement severely limits the range of duty cycles that these protocols can operate at under any circumstance.

- Searchlight does not limit the set of values for t directly even in the asymmetric case. For example, consider a network where nodes can operate at four different duty cycles. To accomplish this we need to chose t_{base}, corresponding to the highest allowable duty cycle. Based on t_{base}, one possible set of allowable values of t can be $\{2 \cdot t_{base}, 4 \cdot t_{base}, 8 \cdot t_{base}\}$, where $8 \cdot t_{base}$ would correspond to

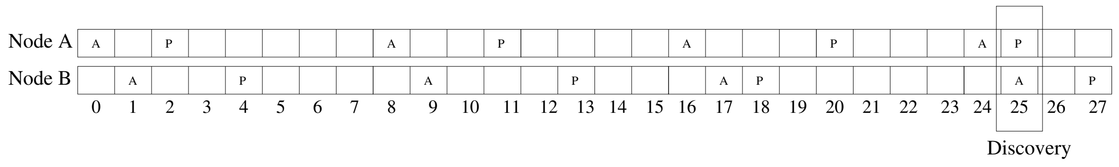


Figure 3.6: Overlap with sequential probing (t=8)

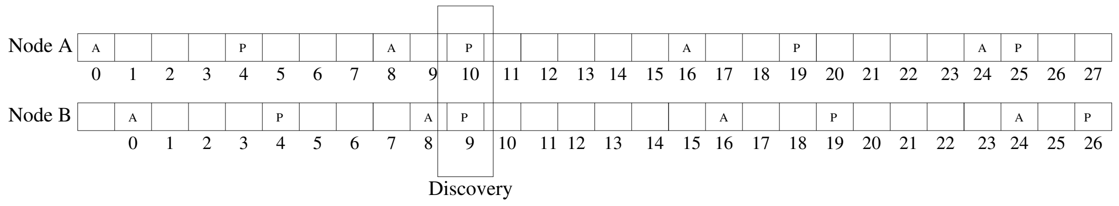


Figure 3.7: Overlap with randomized probing (t=8)

the lowest allowable duty cycle. Now, based on network requirements, the value of t_{base} can be any integer, including non-primes.

Searchlight uses the second approach since it provides a better bound on worst-case latency. However, if needed, both approaches can be employed in the same network. For example, when 4 different duty cycles are needed, three allowable t values can be $t_{base}, 2 \cdot t_{base}, 4 \cdot t_{base}$ and the 4th value can be any *prime* other than 2 that is relatively prime with t_{base}.

The worst-case bound for the second approach is based on the time needed for A's probe slot to meet an anchor slot of B. A's probe slot probes every other slot from 2 to $\lceil \frac{\lfloor \frac{t_A}{2} \rfloor}{2} \rceil$. Let us assume that within that range, there are m anchor slots of B. Let us first consider the case of sequential probing. In the worst-case, when A meets B, A's probe slot would be just past the $m-th$ anchor slot. To overlap, the probe slot will still have to go up to $\lceil \frac{\lfloor \frac{t_A}{2} \rfloor}{2} \rceil$, and then start from 2 again and reach the 1st anchor slot of B. Without striping, this would require the probe slot to travel $t_B + (\lceil \frac{\lfloor \frac{t_A}{2} \rfloor}{2} \rceil \mod t_B)$ slots. With striped probing, the requirement would reduce to $\frac{t_B + (\lceil \frac{\lfloor \frac{t_A}{2} \rfloor}{2} \rceil \mod t_B)}{2}$ periods, or $\frac{(t_B + (\lceil \frac{\lfloor \frac{t_A}{2} \rfloor}{2} \rceil \mod t_B))}{2} \cdot t_A$ slots.

The worst-case bound for sequential probing is based on the observation that A's probe slot is guaranteed to meet an anchor slot of B every t_B periods. However, no such guarantee can be provided for randomized probing. To better understand this point, let us divide the slots of A into t_B buckets. Assuming that the anchor slot is at position 0, the buckets would be as follows:

Bucket 1: $\{1,\ 1 + t_B,\ 1 + 2 \times t_B,\ ...\ \}$

Bucket 2: $\{2,\ 2 + t_B,\ 2 + 2 \times t_B,\ ...\ \}$

....

Bucket t_B: $\{t_B,\ 2 \times t_B,\ 3 \times t_B,\ ...\ \}$

Based on the offset between A and B, B's anchor slots would fall into any of the above t_B buckets. For example, if the offset between A and B is 3 slots, then B's anchor slots with respect to A's anchor slot would be at slots 3, $3 + t_B$, $3 + 2 \times t_B$ and so on (Bucket 3). With sequential probing, it is guaranteed that the movement pattern followed by the probe slot will contain one slot from each of the above buckets every t_B slots. Unfortunately, with randomized probing, no such guarantee can be provided. We next present a slightly restricted form of randomized probing that addresses this issue.

Restricted Randomized Probing in the Asymmetric Case

The basic goal of restricted randomized probing is to ensure that only those movement patterns would be chosen for probing where there is exactly one slot from each bucket every t_B slots. We achieve this goal with two-step randomization. First, a random permutation of the buckets gets chosen. This determines the order in which slots from different buckets would appear. The next step is to randomly choose a permutation of slots within each bucket. We illustrate the strategy with the following example.

Let us assume that $t_A = 60$ and $t_B = 10$. With sequential probing without striping, A's probe slot will follow the pattern (left to right, top to bottom):

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

Each column represents one bucket. The first step of restricted randomized probing is to randomize the order of these buckets. One such valid ordering can be:

6	9	1	10	2	5	7	8	4	3
16	19	11	20	12	15	17	18	14	13
26	29	21	30	22	25	27	28	24	23

The second and final step is to randomize the order of slots within each bucket:

16	9	1	10	22	15	7	18	4	23
6	29	11	20	12	25	17	8	24	3
26	19	21	30	2	5	27	28	14	13

The above pattern ensures that there is exactly one slot from each bucket every t_B slots. Thus, by adopting this approach, restricted randomized probing achieves the same worst-case bound as sequential probing in the symmetric case.

3.2.7 Duty Cycle

So far we have looked at the worst-case bounds on discovery latency provided by Searchlight for different scenarios. But it is important to consider the cost of achieving a particular bound. Since the primary cost we are concerned about is energy, we can look at the duty cycle. For a given t, there are two active slots every t slots. Taking into consideration the cost of extending the slots by δ, the duty cycle for any given t is $\frac{2\cdot(1+\delta)}{t}$. Table 3.1 summarizes the cost (duty cycle) and performance (worst-case bound) of four main deterministic protocols. Searchlight without striped probing has been labeled as NoStripe. However, based on that information, it is hard to compare the different protocols since they use different parameters.

Protocol	Parameters	Duty Cycle	Worst-case bound Symmetric case
Disco	p_1, p_2	$\frac{p_1+p_2}{p_1\cdot p_2}$	$p_1 \cdot p_2$
U-Connect	p	$\frac{3p+1}{2p^2}$	p^2
NoStripe	t	$\frac{2}{t}$	$t \cdot \frac{t}{2}$
Searchlight	t	$\frac{2\cdot(1+\delta)}{t}$	$t \cdot \lceil \frac{\lfloor \frac{t}{2} \rfloor}{2} \rceil$

Table 3.1: Deterministic Protocols

To have a common baseline, let us keep the duty cycle fixed so that all protocols consume energy at the same rate and then look at the worst-case bound given by different protocols. By doing so, we can compare the performance of different protocols when they incur the same cost. Disco does best in the symmetric case when the two primes p_1 and p_2 are as close as possible. For ease of analysis, let us assume that $p_1 = p_2 = p$. Next, we express the parameters of each protocol as functions of x, where $\frac{1}{x}$ is the common duty cycle.

For Disco, $\frac{p_1+p_2}{p_1\cdot p_2} \approx= \frac{p+p}{p\cdot p} = \frac{1}{x}$ and hence, $p = 2x$. Similarly, for Searchlight, $t = 2x(1+\delta)$. Finally, for U-Connect, $\frac{3p+1}{2p^2} = \frac{1}{x}$ and so $p \approx \frac{3x}{2}$. Now we can express the worst-case bounds for all protocols as $f(x)$ (see Table 3.2). It is clear from the table that even with a conservative value of $\delta = 0.1$ ($\frac{1}{10}$-th of the actual slot size), Searchlight reduces the worst case bound by around 50% in comparison to U-Connect, the current best performing deterministic protocol.

3.2.8 Effects of Clock Drift

Searchlight's systematic probing relies on the assumption of a constant offset between anchor slots of any two nodes. In reality, clocks drift and the offset between anchor slots will change over time. However, if the drift is sufficiently small, the worst case bound will not change.

Protocol	Worst-case bound(slots)
Disco	$4x^2$
U-Connect	$\frac{9x^2}{4}$
NoStripe	$2x^2$
Searchlight	$2x(1+\delta))\lceil\frac{\lfloor\frac{2x(1+\delta)}{2}\rfloor}{2}\rceil \approx x^2(1+\delta)^2$

Table 3.2: Worst-case bounds for different protocols operating at the same duty cycle

To understand the effect of clock drift on the worst case bound, let us consider the worst case for two node's schedules. We will assume that no probe slot to probe slot discoveries occur and concentrate on probe slot to anchor slot discovery. It has previously been proven in subsection 3.2.3 that this kind of discovery must exist given any two schedules. The least amount of overlap will occur between the anchor slot and probe slot when the anchor slot starts immediately following the probe slot. Due to the striped nature of Searchlight, the anchor slot of size $x(1+\delta)$ will start at the same time as the inactive slot of size $x(1-\delta)$ producing a total overlap of $x \cdot (2\delta)$ with the next active slot.

Since this anchor slot to probe slot overlap is the only overlap that our worst case analysis relies upon, if the anchor slot were to drift where it was no longer overlapping with this probe slot within one hyper-period, discovery could not be guaranteed. Thus the clock drift must be less than $\frac{x \cdot (2\delta)}{T}$ in order for the worst case bound to hold. Given this constraint, it is easy to choose δ based on the clock drift for a given device such that the bound holds.

3.3 Evaluation

The primary goal of our evaluation is to show that Searchlight achieves significant performance improvements over other asynchronous neighbor discovery protocols by virtue of its systematic probing. Specifically, we evaluate how long it takes for different protocols to discover neighbors when *they spend the same amount of energy*, i.e., operate at the same duty cycle. The time from when two nodes first get in each other's transmission range to the time when they actually discover each other is known as the *discovery latency.* We look at the CDF of discovery latencies to understand the overall trend of a protocol. Since absolute latency in terms of time units is dependent on slot-width and slot-width is platform dependent and the same for all protocols, we use number of slots as the unit for latency. We compare the performance of our approach with the deterministic protocols, Disco and U-Connect, and with the probabilistic Birthday protocol.

3.3.1 State-based Simulation

For any random pair of nodes, the discovery latency can widely vary based on the relative offset. To fully characterize the performance of the different protocols, including the worst-case bound, it is essential to consider all possible offsets through a state-based simulation.

Except for the Birthday protocol, all other protocols basically follow a discovery schedule to determine when to sleep and when to wake up. This schedule repeats every T slots, which we call the hyper-period. When a node is in a particular slot in its schedule, that slot index can be considered the *state* of the node at that point and is always in the range $[0, T-1]$. When two nodes with hyper periods T_1 and T_2 come into each other's transmission range, their states have one of $T_1 \cdot T_2$ possible combinations. For a given combination, the discovery latency is always the same. For Disco, U-Connect and Searchlight-S, we use this observation to loop through all possible combinations and determine the latency for each case. The same approach is not feasible for Searchlight-R since for a given t, a node can choose any of the $(\lfloor \frac{t}{2} \rfloor - 1)!$ patterns to determine the schedule of its probe slot. Instead, we run the protocol 1000 times with different seeds. At each run, we generate a new schedule for both nodes. Then, for that particular schedule pair, we loop through all possible state combinations like we do for Searchlight-S. Since for Disco and U-Connect, all slots are of equal size, we assume the slot boundaries to be completely aligned for the worst case (non-alignment leads to more overlap). On the other hand, since Searchlight uses "overflowing" active slots, complete slot boundary alignment creates the best case scenario. Therefore, we consider the slot boundaries to be offset by half slot-width for simulating Searchlight. For the Birthday protocol, we use a closed form expressions for determining the CDF and the expected value of discovery latency [37].

Symmetric Case

First, we compare the performance of different protocols when nodes operate at the same duty cycle. We look at the cumulative distribution of discovery latencies for all protocols operating at a 5% duty cycle, Disco uses the primes (37,43), Birthday uses probability = 0.5, U-Connect uses the prime 31 and both versions of Searchlight use the integer 40 (see Figure 3.8). To gage the effectiveness of striped probing, we also evaluate both sequential and randomized versions of Searchlight without striped probing. They are denoted by NoStripe-S and NoStripe-R respectively. Both versions of Searchlight *always* achieve the lowest latency in the worst-case, with Searchlight-R doing better in the average case by virtue of its probabilistic component. Without striping, the number of slots needed for discovery becomes double. Still, for 65% of the time, NoStripe-R performs on par with the Birthday protocol or slightly lags behind. Beyond that, the probabilistic nature of the Birthday protocol leads to a long tail and NoStripe-R achieves the lowest latency. In comparison to U-Connect, NoStripe-R achieves better latency all along. For both versions of Searchlight, maximum discovery latency is 400 slots, followed by 800 slots for NoStripe versions. U-Connect comes closest with a maximum discovery latency of 961 slots, which is more than 100% increase over Searchlight.

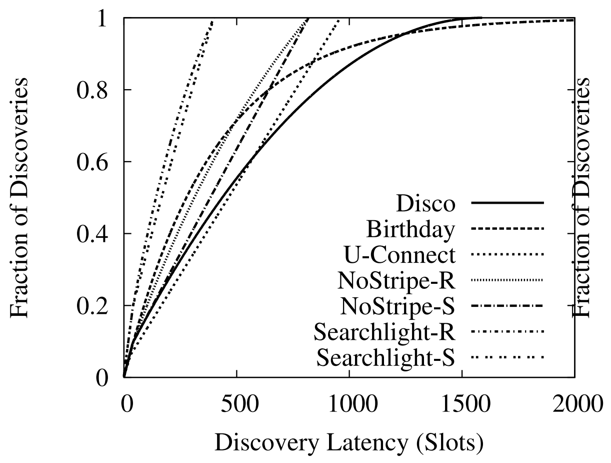


Figure 3.8: State-based simulation: 5% duty cycle

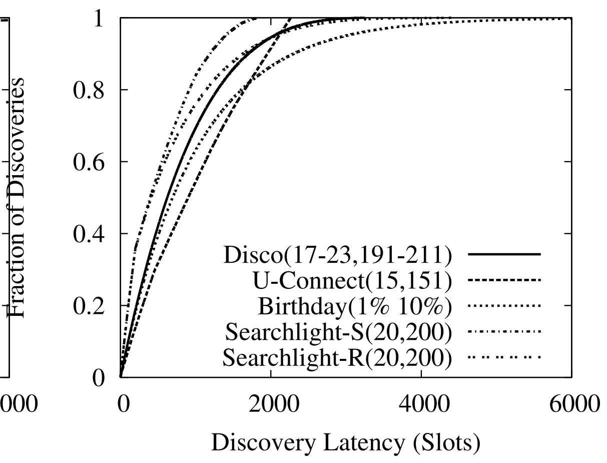
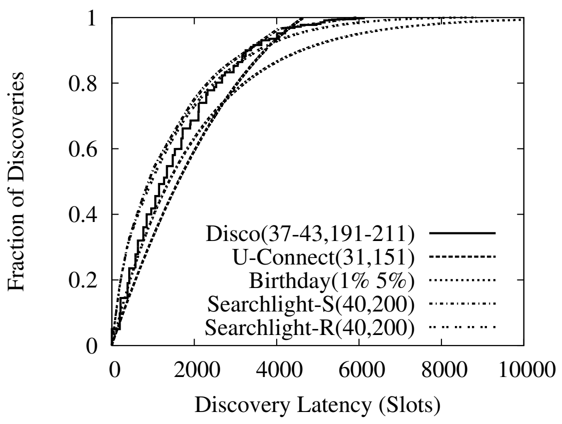


Figure 3.9: State-based simulation: 1%-10% duty cycle

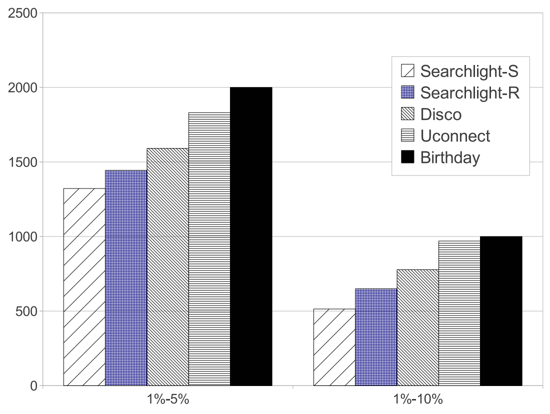
Asymmetric Case

The evaluation of the asymmetric case is particularly important because the worst-case latency is different for Searchlight in this scenario. When two nodes operate at 1% and 10% duty cycles respectively (see Figure 3.9), Searchlight-S performs best all along and its worst-case discovery latency is almost half of U-Connect. Searchlight-R initially performs similar to Searchlight-S, but because of the higher worst-case bound starts to deviate after around 65-th percentile. For the asymmetric scenario when the two duty cycles are 1% and 5% (see Figure 3.10(a)) U-Connect outperforms Searchlight-S in terms of worst-case discovery latency. In the asymmetric case, we know the worst-case bound for Searchlight-S is $\frac{(t_B+(\lceil\frac{\lfloor\frac{t_A}{2}\rfloor}{2}\rceil \mod t_B))}{2} \cdot t_A$, where t_A and t_B are the two periods and $t_A > t_B$. For the 1%-10% case, $\lfloor\frac{t_A}{2}\rfloor = 100$ was completely divisible by the smaller period $t_B = 20$, resulting in lowest worst-case latency. But for 1%-5%, 100 is not completely divisible by $t_B = 40$, which has led to the higher worst-case bound. To further investigate this issue, we look at average latency for both the scenarios (see Figure 3.10(b)). In both cases, Searchlight-S performs best, followed by Searchlight-R, while U-Connect performs even worse than Disco. This shows that in terms of average latency, Searchlight-S still performs best in all asymmetric scenarios.

We have not shown results for energy separately because performances of all the protocols were compared when they operated at the same duty cycle. Since duty cycle directly translates to energy, all our results show how different protocols perform when they use the same amount of energy.



(a) CDF of Discovery Latency 1%-5%



(b) Average Discovery Latency

Figure 3.10: State-based simulation

Effect of restricting Permutation

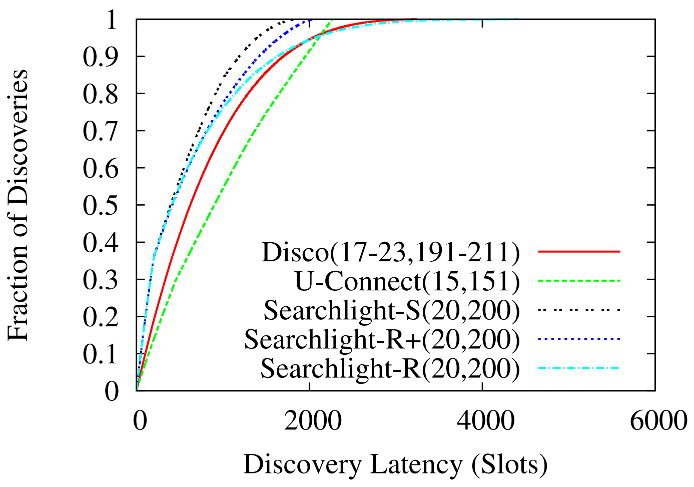


Figure 3.11: Effect of Restricted Randomized Probing: 1%-10% duty cycle

We next evaluate the effect of restricting the set of allowable permutations for randomized probing. When two nodes operate at 1% and 10% duty cycle respectively, the worst-case latency for Searchlight-S is less than 50% of the worst-case latency for Searchlight-R (see Figure 3.9). But when we restrict the set of movement patterns that can be chosen (see Section 3.2.6 for discussion), the worst-case performance of such restricted randomized probing (Searchlight-R+) becomes almost the same as Searchlight-S (see Figure 3.11).

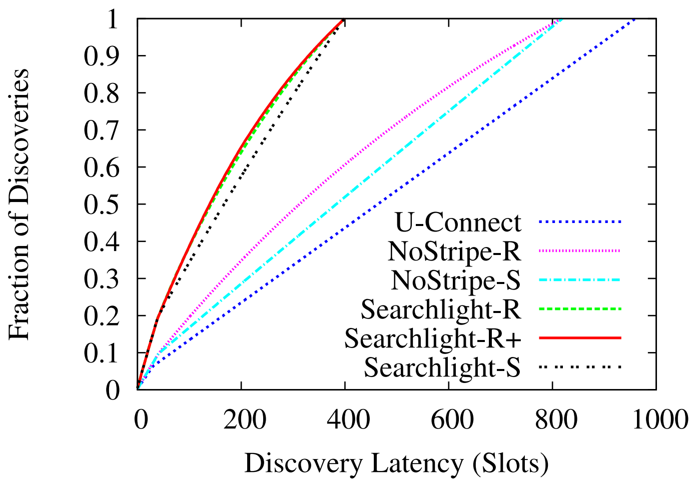


Figure 3.12: Effect of Restricted Randomized Probing: 5% duty cycle

While such restricted randomization helps in the asymmetric case, it is important to evaluate whether it decreases the advantage of Searchlight-R in the symmetric case. We next look at the comparative performance of Searchlight-R+ when all protocols operate at 5% duty cycle. Since 5% duty cycle corresponds to a period of 40 slots for Searchlight, we use 10 slots as the base period for restricting randomization. As we can see from Figure 3.12, Searchlight-R+ performs the same as Searchlight-R. This shows that the reduction in randomization does not affect the average-case behavior of randomized probing in the symmetric case.

3.4 Implementation

One of the main objectives for designing asynchronous neighbor discovery protocols is to facilitate ad hoc communication between handheld devices like smartphones. To gauge how Searchlight achieves this goal in practice, we implemented the protocol in Python and deployed it on the Nokia N900, a smartphone by Nokia that supports the open source Maemo 5 mobile device platform [5], and a Dell Latitude E6400 laptop with an Intel 5100AGN Wi-Fi card. Results shown are collected strictly from the Nokia devices. We first describe different implementation issues and then present some preliminary results.

3.4.1 Implementation Issues

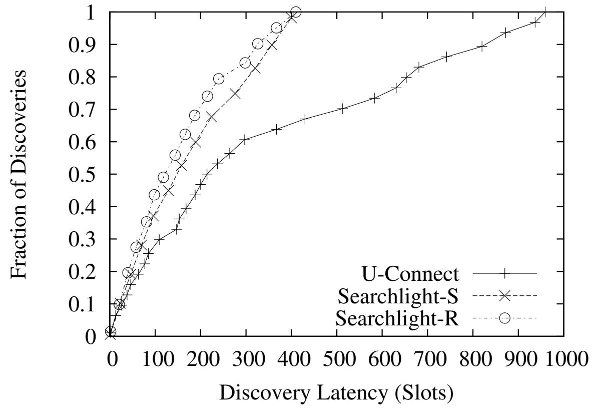
- **Slot Duration** We implemented Searchlight to use the Wi-Fi radio of the N900 phone for neighbor discovery. Earlier protocols for asynchronously discovering neighbors were all implemented on sensor nodes, allowing them to use small slots on the order of milliseconds [22] or even microseconds [29]. However, unlike sensor radios, Wi-Fi radios have a non-negligible transition latency from sleep to transmit/receive. On the N900 phone, we found out that from the application level, the time to bring the wireless interface up is around 1 second. Because of this latency, we decided on a slot size of 2 seconds. Other phones we have looked at, including the Android G1, also have startup latencies on the order of seconds.

- **Pre Slots** Because of the non-negligible start-up time, we introduced the notion of *pre* slots. A *pre* slot basically precedes any active slot and switches on the interface. For example, assume that a node needs to be active during slot 10. Now, if the command to wake up the radio is issued at the beginning of slot 10, it will take almost the whole slot duration for that command to return and the effective awake time during that slot will be reduced to 1 second. To get around this problem, the wake up command now gets issued at the start of the preceding slot which is slot 9 in this case. Such slots are called *pre* slots and their positions are determined based on the active slot schedule. *Pre* slots are kept shorter than normal slots to allow for the larger active slots. In U-Connect, two active slots can be neighbors. In this case, the radio is left on rather than using a *pre* slot.

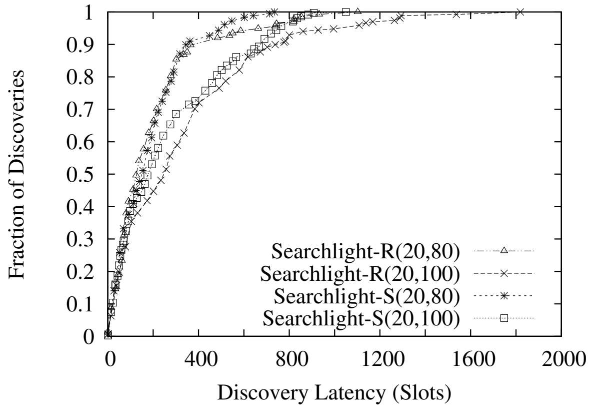
- **Active slot** When the protocol starts up, it creates an active slot schedule based on the desired duty cycle. In an active slot, a hello message containing node id gets sent at the very beginning and at the very end of the slot. In between, the node continuously listens for hello messages from other nodes. When it gets a hello message, it adds the name of the sender to a discovered list. The size of the slot extension for active slots, which guarantees slot overlap, was found to be 5 milliseconds. This more than accommodates a hello message transmission and jitter from timer scheduling.

3.4.2 Evaluation

We implemented Searchlight-S and Searchlight-R on five N900 phones and logged the discovery latency for around 200 runs with 5% duty cycle. We also implemented U-Connect since it performs best among existing protocols in the symmetric case. Between the implementations, Searchlight-R and Searchlight-S fare much better than U-Connect, similar to our simulation results (see Figure 3.8). When striped probing is added to U-Connect, its performance increases but was still found to fall short of Searchlight. We also looked at the asymmetric case of Searchlight with t values of (20,80) and (20,100) which represent 10%, 2.5%, 10%, and 2% duty cycles respectively. Searchlight-S outperforms its randomized version as was predicted in the asymmetric case.



(a) Symmetric



(b) Asymmetric

Figure 3.13: Implementation: CDF of Discovery Latency

3.5 Conclusions and Future Directions

Solving the problem of energy efficient asynchronous neighbor discovery is an important precondition for successful widespread adoption of mobile opportunistic networking. In this chapter, we presented Searchlight, a new asynchronous neighbor discovery protocol that addresses this problem by adopting a systematic probing based approach to provide better bounds on discovery latency than any existing protocol when nodes have similar energy requirements. Extensive simulation results show that Searchlight achieves the best average case discovery latency in most of the scenarios and performs on par with other protocols in the remaining cases. Searchlight was also successfully implemented on a smartphone testbed, and showed performance trends similar to the simulation results.

The problem of finding the appropriate value for δ needs to be further investigated. Setting it too low would make the execution of the protocol more vulnerable to clock drift and interference. On the other hand, setting the value of δ to a large value will increase the discovery latency for a given duty cycle. It would be also useful to investigate how Searchlight or any other discovery protocol can dynamically adapt to energy requirements, contact patterns and other factors to adjust its duty cycle. For example, when neighbor count is low, a higher duty cycle might be required to find enough neighbors that meet application requirements. On the other hand, in a crowded place where the number of co-located nodes is high, a lower duty cycle might suffice to find a reasonable number of neighbors.

At a high level, it would be interesting to figure out whether average-case performance or worst-case performance is more important for opportunistic communication. This is important since picking the right neighbor discovery protocol often requires making a trade-off between these two objectives.

Chapter 4

Extending the reach of Neighbor Discovery through Co-operation: *CQuest*

Devices that participate in opportunistic communication often come equipped with multiple radios, especially in pocket-switched networks. For an individual node, performing neighbor discovery can be too expensive with a high-power (HP) radio like Wi-Fi. As a result of this expense, most distributed mobile social networking applications limit their discovery to the use of a low-power (LP) radio like Bluetooth (e.g.,Mobiclique [52], PeopleNet [38]), significantly reducing energy costs. However, this reduction comes at the cost of a significant reduction in the discovery of neighbors, and so contact opportunities, due to the lower range of LP radios. The ultimate result of dismissing the use of the HP radio is degraded application performance and unsatisfied users. To mitigate this problem, we have developed CQuest, a novel scheme for more efficient HP neighbor discovery that leverages the clustering of nodes as well as the radio heterogeneity of mobile devices. The basic idea is that coordination over a LP radio can help clustered nodes distribute the load of HP scanning. We present results from extensive simulation that shows CQuest discovers significantly more contacts than a low-power only scheme but without incurring the high energy cost usually associated with long-range discovery. We have also implemented the protocol on a testbed of Android G1/G2 phones that shows the feasibility of the protocol in a real network.

4.1 Exploiting Radio Heterogeneity & Clustering

Mobile peer-to-peer networking among geographically co-located nodes has the potential to expand the reach of many social networking and mobile gaming applications. However, the success of such support depends on a node's ability to discover its neighbors in an energy efficient manner. Additionally, the availability of multiple radios (e.g., Bluetooth, Zigbee, Wi-Fi, etc.) makes this problem of energy-efficient discovery all the more challenging, yet also more achievable. In this section, we discuss the importance of radio heterogeneity for neighbor discovery and the challenges to making it energy efficient.

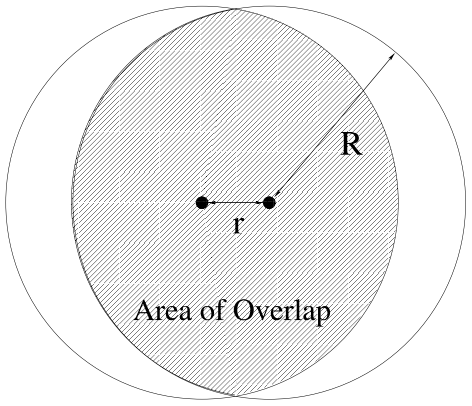


Figure 4.1: Area of Common HP Neighbors

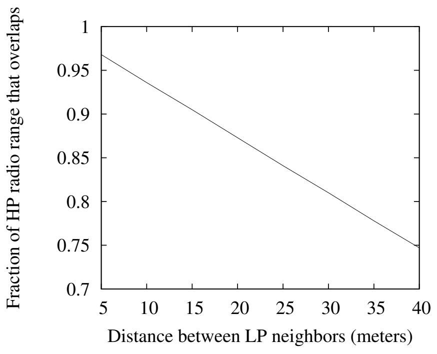


Figure 4.2: Probability of a common HP neighbor

4.1.1 Discovery in Multi-radio Networks

Most mobile communication devices now come equipped with multiple wireless interfaces (e.g., Bluetooth, Wi-Fi) that significantly vary in terms of energy-efficiency and transmission range. This range of services and costs presents interesting trade-offs when considered in the context of neighbor discovery. LP radios like Bluetooth consume significantly less energy during discovery, but at the cost of considerably reduced communication ranges. On the other hand, an HP radio (e.g., Wi-Fi) can achieve around a ten-fold increase in terms of transmission range, but the cost of discovery becomes prohibitively high (e.g., Mobiclique [52]).

This trade-off was realized very early on and many of the original multi-radio schemes focused on energy conservation at the cost of reduced range. By focusing on neighbors that are reachable through *both* radios, the LP radio can be used by default [12] and the HP radio is only turned on when it is needed by high bandwidth applications such as VOIP [59, 9]. For general applications, the LP radio can be used as a kind of pager [10, 32] or help with device discovery and connection setup [49]. Radio heterogeneity has also been leveraged for discovering Wi-Fi access points in an energy-efficient way. For example, Blue-Fi [11] uses Bluetooth contact patterns and cell-tower information to predict the availability of Wi-Fi access points and hence save on the energy cost of scanning for APs using the Wi-Fi radio. Another recent system, Zi-Fi [72], uses Zigbee radios instead of Bluetooth for discovering Wi-Fi access points. Along with control signaling, CoolSpots [48], Turducken [60] and SwitchR [8] also use the LP radio for data communication. While providing significant energy savings, most of these dual-radio approaches ignore the asymmetry in transmission ranges and so reduce the effective range of the HP radio to that of the LP one.

4.1.2 Extending the Range

More recent protocols have started looking at the possibility of judiciously using the HP radio (i.e., Wi-Fi) to reach nodes that are not directly reachable through the LP radio. However, these protocols either completely ignore the need for discovery or do not make any effort to reduce the high cost of Wi-Fi discovery and scanning. For example, the bulk communication protocol (BCP) [58] sends control messages over multiple hops with the short-range radio to enable single-hop long-range (Wi-Fi) communication. However, BCP is aimed at relatively static sensor networks and so assumes that each node learns its HP and LP neighbors at configuration time and so does not consider discovery costs as part of its optimizations. On the other hand, CONET [67] is a distributed clustering protocol that selects a cluster head for each Bluetooth cluster that acts as the gateway between the cluster and a Wi-Fi access point in an infrastructure network. While nodes running CONET cooperate to reduce the cost of data transmission, all nodes still pay the high cost of discovering Wi-Fi access points individually. Essentially, while these protocols effectively collaborate to reduce energy consumption for data transmission, they both ignore the cost of Wi-Fi scanning and discovery.

4.1.3 Clustering and Collaboration

In a mobile network, collaboration is a natural approach to both sharing information and reducing energy costs. Essentially, the mobility of nodes in real life often leads to the formation of temporary clusters [65, 27] that open up new opportunities for collaboration. For example, consider a group of people with smartphones using GPS to support location-based services. Since GPS is one of the most energy-hungry components of a smartphone [46] and most applications do not need perfect location information, near-by nodes could cooperate so that instead of all nodes using their GPS simultaneously, each node could take turns and share the obtained GPS information.

Such cooperation can also be used to reduce the high cost of neighbor discovery for multi-radio mobile nodes. Essentially, the goal is to allow the nodes to take turns doing the expensive scanning and discovery over the HP radio and then share the information over the LP radio. In this context, the goal is to enable all nodes to discover both *LP* neighbors over the low-power radio and *HP* neighbors over the HP radio.

While the use of clusters may lead to some inefficiencies, it is interesting to note that two LP neighbors typically have quite a few HP neighbors in common. With two LP neighbors, A and B, when B scans for its HP neighbors, B will discover most of A's HP neighbors as well and so can share this information with A.

To better understand the extent of overlap, let us consider a simple scenario consisting of two LP neighbors, A and B. For simplicity, assume the LP and HP transmission ranges are unit disks with radius r and R respectively (see Figure 4.1). Assuming that A and B are separated by the maximum distance r, the intersection of their HP radio ranges is given by the following expression:

$$A_{intersection} = 2 \cdot R^2 \cdot acos(\frac{r}{2 \cdot R}) - \frac{1}{2} \cdot r \cdot \sqrt{4 \cdot R^2 - r^2}.$$

So, assuming nodes are uniformly distributed in space, the probability P that a node reachable by A through A's HP radio is also reachable by B through B's HP radio is given by the following equation:

$$P = \frac{A_{intersection}}{A_R}. \tag{4.1}$$

For typical transmission ranges of Wi-Fi and Bluetooth, $R = 100$m and $r = 10$m respectively, the value of P is 0.936. Essentially, for any pair of Bluetooth neighbors, there is very high probability that the two nodes will have the same set of Wi-Fi neighbors even when they are separated by the *maximum* Bluetooth distance (see Figure 4.2). This shows us that while not perfect, collaboration can provide quite effective discovery if it can be designed to be energy efficient.

While collaboration has been explored in the context of mobile social networks, all current approaches are limited to finding Wi-Fi access points in networks with stable peers. For example, Blue-Fi [11] has a collaborative predictive component that allows Bluetooth peers to exchange information for enhancing the prediction process that determines when the Wi-Fi interface should be switched on to scan for APs. MOBIX [47] also relies on information exchange over Bluetooth between mobile peers for energy-efficient network selection. In the context of more dynamic and single-radio networks, Disco [23] proposed that nodes can exchange their neighbor lists to speed up the discovery process, although it was not investigated in the paper. However, none of these protocols have explored the possibility that nearby nodes can actually share the load of discovery instead of just the results of discovery. In response, we next present CQuest, a collaborative protocol that goes beyond these limitations and effectively leverages both clustering and radio heterogeneity for performing *neighbor discovery* over the HP radio at significantly reduced energy costs, which ultimately leads to more effective communication in mobile social networks.

4.2 CQuest

The overarching goal of CQuest is to reduce the energy cost of HP neighbor discovery by leveraging clustering and radio heterogeneity. Because of the range disparity between HP and LP radios, the nodes within an LP neighborhood are very likely to discover the same set of HP neighbors. Given this redundancy, not all nodes in the network need to discover HP neighbors at the same time. Instead, HP discovery can be performed by a *subset* of nodes, which then disseminate the resulting information to non-discovery nodes. Since discovery is performed by scanning the wireless channel, we call this the *scanner set*, S. To enable high-probability discovery of HP neighbors, the nodes in the scanner set should provide coverage of the union of the HP neighborhoods of the cooperating nodes. Additionally, to load balance energy consumption, S should change over time so that the load gets distributed as evenly as possible. In this section, we first formalize the scanner set selection problem and discuss why it is hard to achieve an ideal solution in a real network. We then describe the CQuest protocol and how it approximates the goals of coverage and fairness in a practical way.

4.2.1 Scanner Set Selection

To enable cooperation, nodes must determine whether or not they should be scanning and discovering on the high-power Wi-Fi radio. To understand how to optimize the selection of scanners to achieve both good energy efficiency and effective discovery, consider a network of nodes, which we represent as an undirected graph $G = (V, E)$, where V is the set of $N = |V|$ vertices, and E is the set of links via a low-power radio. Each node is also equipped with a high-power radio. Let $W(x)$ denote the set of all nodes that any node x can discover if it scans for neighbors using its high-power radio. Let S, which we call the *scanner set*, denote the set of nodes chosen to perform high-power scanning.

Theorem 4.2.1. *Any set S provides* discovery coverage *to graph G with vertex set V if the following holds:*

For any arbitrary node i,

$$i \in \bigcup_{x \in V} W(x) \Rightarrow i \in \bigcup_{y \in S} W(y)$$

Intuitively, the above definition means that a subset S of V provides discovery coverage to G if the area reachable by the high-power radios of the members of S is no less than the area reachable by all of the nodes in G. That means that even if nodes belonging to $V - S$ do not perform high-power scanning, it does not reduce the chance of discovering any arbitrarily placed node. For example, in Figure 4.3, $S = \{A, C, D, E\}$ provides discovery coverage to the graph since omitting B does not decrease the total area reachable through the high-power radios.

However, finding such a set S requires the knowledge of the exact positions of all the nodes, as well as complex geometric calculations. Additionally, to achieve energy savings, the cardinality of the set S should be minimal, which introduces additional complexity to the problem. So, instead of finding the exact discovery coverage, we propose finding an approximate solution by making the following assumption:

For any two nodes u and v,

$$\{u, v\} \in E \Rightarrow W(u) = W(v)$$

This assumption satisfies coverage if the set of high-power neighbors that can be discovered by two lower-power neighbors are same. While this is only true if the two nodes are located at the exact same position, we showed that there is a high probability that coverage will still be satisfied even if the low-power neighbors are separated by the maximum distance (see Section 4.1.3). With this assumption, the problem of constructing the set S becomes the same as finding a dominating set D of V such that every vertex not in D is joined to at least one member of D by some low-power edge. To reduce the overall energy cost, the cardinality of the dominating should again be minimal.

However, picking one minimum dominating set will lead to unfairness and unequal energy costs across the nodes. Essentially, if the same set of nodes always scans, their energy will drain quickly. Instead, the cost of high-power discovery needs to be shared across the low-power cluster. To facilitate sharing, we consider a discrete-time model, where time is divided into rounds of equal length. For each round t, S_t denotes the scanner set for that round. Let a configuration $c = \{S_1, S_2, ..., S_t, ...S_n\}$ be a set of scanner sets for $n = |c|$ rounds, and C is set of all possible configurations. Let $b_i(t)$ be a binary variable which is set to 1 if node i is a member of set S_t, and 0 other-wise. We can now define the load $L_i(c)$ of a node in a particular configuration c in the follow way:

$$L_i(c) = \sum_{t=1}^{n} \frac{b_i(t)}{n}.$$

To share the energy cost across all nodes, we adopt the idea of min-max fairness [7]. Formally, min-max fairness is defined as follows.

Theorem 4.2.2. *Min-max fairness: A configuration x is min-max fair if it is not possible to decrease the load of a node i, $L_i(x)$, without increasing the load of some node j, $L_j(x)$ $(i \neq j)$ with $L_j(x) \geq L_i(x)$*

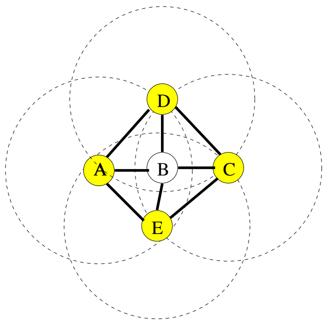

D
A
B
C
E
solid lines represent Bluetooth link
dotted lines represent Wi–Fi range

Figure 4.3: Discovery Coverage of a Graph

in any other configuration y:

$$(\exists i \in V) L_i(y) < L_i(x) \Rightarrow (\exists j \in V) L_j(y) > L_j(x) \geq L_i(x)$$

The objective then is to find a configuration c for G such that,

1. For any $S \in c$, S should be a minimal dominating set of G,

2. c should be min-max fair as described in Definition 4.2.2.

The first condition ensures discovery coverage (based on the assumption previously stated) without having any redundancy, while the second condition ensures that the load of scanning gets distributed throughout the network.

However, even with global knowledge, computing a single minimal dominating set of a graph is NP-complete [24]. So it is obvious that finding a set of minimal dominating sets that satisfies the fairness criteria would be all the more infeasible. Also, in a mobile wireless network, not only is global knowledge unavailable to individual nodes, changing topology renders use of an exact but time-consuming algorithm ineffective. Therefore, we next present CQuest, a very simple distributed coordination protocol that uses a heuristic-based approach that ensures coverage and achieves fairness while reducing redundant scanning.

4.2.2 The Protocol

CQuest has three major components: scanner set selection, which enables cooperative discovery of HP neighbors at reduced energy costs, HP and LP scanning, which provides the actual discovery of neighbors, and neighbor maintenance, which enables nodes to exchange the results of HP scanning over the

LP radio so that non-scanner nodes can learn about their potential HP neighbors without scanning themselves.

- **Scanner Set Selection:** The main challenge for scanner set selection is providing a light-weight distributed solution to probabilistic coverage in a dynamic network. In a static network, S_t can be determined during network set up. However, in a network with mobile nodes, S_t needs to be updated periodically to maintain coverage. Additionally, composition of S_t needs to change periodically to distribute the load of scanning. Given these requirements, CQuest assumes a discrete-time model, where time is divided into rounds of equal length and S_t is determined for each round. The exact definition of a round in time units can be determined based on actual network characteristics and is not tied to the functioning of the protocol.

  The goal of scanner selection in CQuest is to construct S_t for each round, where S_t is dominating for that round. Due to the dynamic nature of mobile networks and need to reduce control overhead, CQuest takes a purely local approach that enables each node to independently determine whether or not it should be a member of S_t based on simple contention-based approach. The result is an approximation of a dominating set that errs on the side of a small amount of redundancy in S_t.

  In CQuest, each node determines its own membership in S_t by looking at its LP neighbors. Only if none of its LP neighbors is in S_t, the node includes itself in the scanner set. This type of approach is analogous to MAC-level contention resolution in multi-hop wireless networks. In MAC protocols, nodes use carrier sensing (physical, and sometimes virtual) to see if any other node in the vicinity is transmitting. If not, it proceeds to transmit. Otherwise, it waits for the transmission to end. Similarly, the basic idea for scanner selection in CQuest is that a node checks to see if any of its LP neighbors is going to do a scan for HP neighbors in the next round (i.e., is a member of S_t). If yes, the node does not scan in the next round to reduce redundancy. Otherwise, the node starts scanning in the next round.

  To reduce the possibility of collisions or redundant scanning, CQuest uses a contention phase at the end of each scanning round. The contention phase is divided into contention slots, where the number of slots is a configurable protocol parameter called *windowSize*. At the beginning of each contention phase, nodes randomly pick a slot between 0 and *windowSize* and broadcast a contention packet in that slot over the LP radio. If a node receives a contention packet before its own selected slot, the node decides that is has “lost” that contention phase and one of its neighbors is in S_t. Otherwise, if a node does not get any contention packet before its own selected slot comes

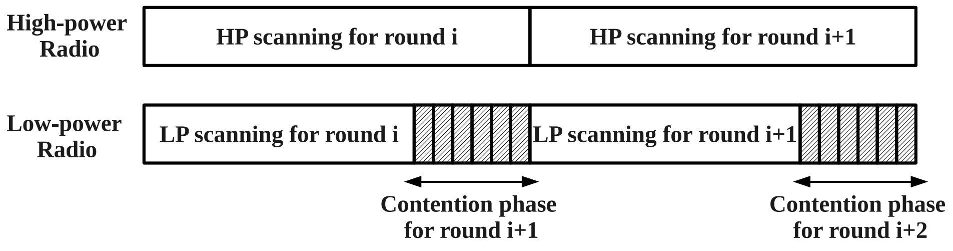

High-power Radio
HP scanning for round i
HP scanning for round i+1
Low-power Radio
LP scanning for round i
LP scanning for round i+1
Contention phase for round i+1
Contention phase for round i+2

Figure 4.4: Contention Phase

up, it broadcasts a contention packet itself and considers itself as the winner of that contention phase and includes itself in S_t.

Obviously, using this type of contention resolution may result in occasional collisions and multiple LP neighbors could end up scanning in the same round. To reduce control overhead and to maintain the completely distributed nature of the protocol, CQuest does not require a minimal dominating set and allows "redundant" scanning. Similarly, CQuest trades off simplicity and reduced control overhead for the guarantee of coverage in the face of mobility. If a scanner node leaves a cluster, there may be a period of time without complete discovery coverage.

The final part of scanner selection is load balancing. If the same set of nodes always scans, their energy will drain quickly. Instead, the cost of HP discovery needs to be shared across the LP cluster. To achieve fairness, CQuest attempts to improve the chances of a node being a scanner in a round if it was not selected in the previous round(s). While different techniques can be applied to achieve this objective, CQuest uses an exponentially decreasing contention window size. Initially, nodes start with a $windowSize$ equal to *max-window-size*. Every time a node fails to win a contention phase, it decreases the $windowSize$ by reducing it by half, until it reaches the *min-window-size*. On the other hand, if the node wins a contention phase, it resets the $windowSize$ to *max-window-size*.

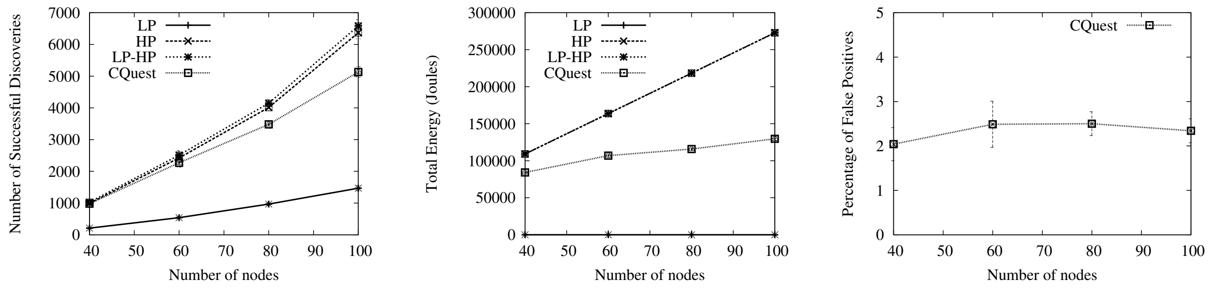
- **HP and LP Scanning:** After selection of the HP scanner set through coordination over the LP radio, the next step is to scan for neighbors using both of the radios. In each round, scanners use their HP radios to discover any node in their HP neighborhood and non-scanners turn off their HP radios. The actual protocol used for scanning and discovery is determined by the specific network environment. In a synchronous network, the discovery protocol can be as simple as using a beacon with a fixed period, and so a round could be defined as some pre-defined number of such periods. For asynchronous networks, existing asynchronous neighbor discovery protocols (i.e., U-connect [29]) can be used. The LP radio can run discovery process of its own anytime except for during the contention phase. Again, the exact method for discovery depends on the radio used (e.g., Zigbee, Bluetooth), application requirements, etc.

- **Maintenance and Exchange of Neighbor Information** CQuest supports direct neighbor discovery (either LP or HP) through beaconing, or indirectly through disseminated neighbor information from other nodes. HP discovery beacons include the IDs of all LP neighbors. Similarly, LP discovery beacons include the IDs of all HP neighbors. Once a node gets added to the neighbor database, subsequent discoveries refresh that information. CQuest uses two thresholds to determine the staleness of an entry based on the last refresh time. If the time elapsed since the last refresh time is more than *freshness-threshold*, the entry still remains in the database but is not included as part of neighbor information exchanged. On the other hand, if the time elapsed since the entry was last refreshed exceeds *expiration-threshold*, the entry gets completely removed from the database. Obviously, *expiration-threshold* $\geq$ *freshness-threshold*.

4.3 Evaluation

The goal of our evaluation is two-fold. First, we evaluate the effectiveness of CQuest in terms of energy consumption and successful neighbor discovery. We show that CQuest finds significantly more neighbors than a simple low-power approach, and saves a significant amount of energy as compared to a simple high-power approach. The second component of our evaluation is to see the impact of the slightly reduced contact opportunities of CQuest as compared to the high-power approach by using a common DTN routing protocol. In this context, CQuest actually out performs the high-power approach in terms of delay and provides comparable delivery ratios.

To evaluate the effectiveness of CQuest, we consider three metrics. First, given the main goal of neighbor discovery, the **Number of Successful Discoveries** captures the effectiveness of a given discovery protocol. Second, **False Positives** captures the effect of indirect discovery, which can lead to discovery of contact opportunities that do not exist. For example, A might first discover B through the high-power radio and then discover C through its low-power radio. A then tells C about B but there may be no contact opportunity between B and C at all. This can happen for two reasons. One is stale information - B may have simply moved from the earlier location by the time A told C about it. Another reason might be lack of overlap. B might be in a location that falls within the transmission range of A's high-power radio but remains out of reach of C's high-power radio. Finally, the **Total Energy** consumed during the use of each protocol determines its ultimate efficiency. For the high-power interface, we include the energy consumed for on-to-off transitions, beacon transmissions, beacon receptions,



(a) Number of Successful Discoveries (b) Energy (c) False Positives

Figure 4.5: Comparison of different discovery schemes

idle time spent during an active slot, and also off-to-on transitions. For the low-power interface, we include the energy consumption of beacon transmissions and receptions. We omit the idle energy consumption for the low-power radio, since the low power interface is on all the time for all protocols.

4.3.1 Mobility model

To see the effect of social mobility and the formation of clusters, we use the community-based mobility model described in [39], which captures user community structure in an implicit way by quantifying the relationship between the nodes of the network. Essentially, each node is associated with some set of nodes that belong to its community. The weight of the edge connecting two nodes signifies the strength of their social relationship. The simulation of this model generates traces with characteristics similar to that of real life traces obtained from Intel Research in Cambridge. We vary the number of nodes from 40 to 100 to study how CQuest performs in networks with different densities.

The protocols were implemented on the ns2 [6] simulator. To simulate dual-radio nodes, we used the NS-MIRACLE extension [14]. However, NS-MIRACLE does not support Bluetooth. Hence, for the short-range radio, we use a radio that has energy, transmission range, and bandwidth characteristics similar to Bluetooth but is capable of broadcasting. We discuss how CQuest can be adapted to work with a regular Bluetooth radio in Section 4.4. For all graphs, error bars denote 95% confidence intervals.

4.3.2 Evaluation Protocols

We compare the following schemes with CQuest:

- **LP**: Nodes use only their low-power radio to detect contacts. The interface is always on and periodic beacons are broadcast to enable discovery. This is the most common single radio approach used by most current mobile social networking applications (e.g., [52, 57]). This approach uses the least amount of energy and provides a baseline for the minimum number of contacts discoverable.

- **HP**: Nodes use asynchronous neighbor discovery over their high-power radios and keep their low-power radios completely off. For our evaluation, discovery was performed using U-Connect with $p = 19$.
- **LP-HP**: Nodes use both radios for neighbor discovery in a combination of LP and HP, but without any coordination-ordination.

4.3.3 Successful Detection of Contacts

We start by comparing the performance of the different schemes in terms of the absolute number of discovered contacts. Specifically, we are interested to see if using a high-power radio actually increases the number of discoveries in practice. For all node densities, CQuest discovers significantly more contacts than the LP scheme, with a minimum improvement of around 250% (see Figure 4.5(a))). On the other hand, in comparison to HP-LP and HP, even in the worst case, CQuest discovers 15%-20% fewer contacts, since it does not guarantee coverage. One interesting thing to note is that the relative performance of CQuest actually drops with increasing node density. This is because the increase in node density creates more opportunity for cooperation. However, nodes in the network are not always static. So it may happen that after a node becomes a scanner in a cluster for a specific round, it moves away. This can cause the non-scanning neighbors of that cluster to miss few long-range contacts completely.

4.3.4 Energy

The cost of using high-power radio for discovery is an increase in energy consumption. Next, we look at how much energy savings CQuest offers in contrast to HP and HP-LP modes that have higher discovery rates. We can see from Figure 4.5(b) that CQuest achieves an energy savings of around 23% when node density is lowest (discovery success rate of CQuest was only around 1%-5% lower for the same density). As expected, the savings increases with increase in node density, and for 100 nodes, the energy savings exceed 50%.

4.3.5 False Positives

The indirect method of discovery in CQuest can lead to false positives about reachability. We calculated the number of such false positives as a percentage of successful discoveries. For all densities, the fraction of false positives remained very small and almost constant in the interval 2%-2.5%. This confirms that two low-power neighboring nodes often share the same high-power neighborhood.

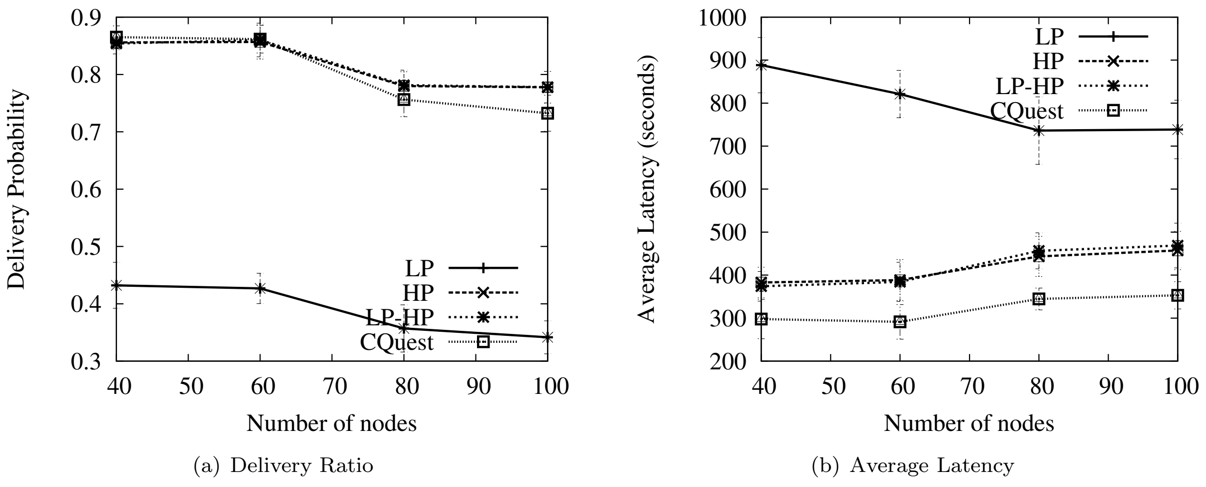


Figure 4.6: Effect of different discovery schemes on the performance of Spray and Wait

4.3.6 Effectiveness of Discovered Contacts

The primary motivation for enabling the discovery of more neighbors is to enhance the performance of protocols that depend on opportunistic contacts. To evaluate the extent to which CQuest achieves this goal, the contact traces from the simulation of the different discovery schemes were given as an input to an opportunistic routing protocol. To compare the impact of different discovery schemes, we looked at two metrics. *Delivery ratio*, the percentage of total generated messages that eventually get delivered, and *Delay*, the average latency per delivered message.

For our simulations, we used the Opportunistic Network Environment (ONE) simulator [31] and chose Spray and Wait [61] as the routing protocol because of its resource friendliness. One message was generated at a randomly selected node every second and the destination for that message was also chosen randomly. For all runs, the message size was varied from 1 KB to 10 KB. Each data point is the average of 10 runs. Since we are interested only in seeing the impact of contacts, the buffer size was set to a very large value (500 MB) so that performances of the protocols do not get affected by buffer constraints.

Simulation Area	$500m \times 500m$
Range of the high-power radio	100 m
Range of the low-power radio	10 m
bt_beacon_period	5 seconds
Protocol used for Wi-Fi Discovery	U-Connect (p=19)

Table 4.1: Simulation Settings

4.3.7 Delivery Ratio

Irrespective of number of nodes in the scenario, schemes that use the high-power radio (i.e., HP, LP-HP, CQuest) deliver many more messages than LP (more than a 100% improvement in delivery ratio for all densities) (see Figure 4.6(a)). However, more important to evaluate is the performance of CQuest in comparison to HP and HP-LP, since, as we have seen earlier CQuest does fail to detect some contact opportunities since, unlike those schemes, all nodes are not scanning all of the time. The delivery ratio indicates the significance of those missed contacts. As we can see, although CQuest discovered fewer contacts than HP and LP-HP, its delivery probability is almost as good as the two protocols. This means that the contacts missed by CQuest were probably too short to be taken advantage of anyway.

4.3.8 Average Latency

For opportunistic networks, including mobile social networks, delivery latency is also very important since messages may lose utility if they take too long to be delivered. As we can see again, the delivery latency of CQuest is significantly lower than LP (see Figure 4.6(b)). On the other hand, in spite of discovering fewer contacts than HP and HP-LP, CQuest actually delivers messages faster, showing that the contacts discovered by CQuest are actually more stable and useful, leading to better latency.

From our evaluation, it is evident that CQuest achieves its goal of increasing the number of discovered opportunities significantly in comparison to a low-power only scheme, while significantly reducing the energy cost of long-range discovery in comparison to non-cooperative high-power radio approaches. Also, we can conclude that though the cooperation leads to slightly lower discovery rate for CQuest in comparison to HP and HP-LP, its ability to find *contacts that actually matter* is on par with those protocols. Such performance parity with the best performing existing schemes, along with significantly less energy consumption, makes CQuest a perfect candidate for being used in conjunction with opportunistic routing protocols.

4.4 Implementation

To evaluate how CQuest performs in a real setting, the protocol was implemented on a testbed of 15 Google G1 and G2 Developer phones running a modified version of CyanogenMod 6.0[1] which supports Android 2.2. While the Android platform could be modified to support the CQuest implementation, the 802.11 drivers could not be modified such that the 802.11 radio could be switched on and off via the

[1]See http://www.cyanogenmod.com/

driver itself. Instead, toggling of the 802.11 radio on and off had to be accomplished by loading and unloading the entire driver module respectively, resulting in a significant delay. However, the biggest challenge of the implementation was adapting CQuest to the non-broadcast medium of Bluetooth.

4.4.1 Modification of CQuest to work with Bluetooth

The original design of CQuest assumed a broadcast capable low-power radio to support the contention phase. To work with Bluetooth, the contention phase had to be modified from implicit coordination via broadcast to explicit coordination via directly exchanged messages. Specifically, with a broadcast medium, nodes receive messages and act on the information they contain without any further communication required. However, Bluetooth requires that a connection be established between two nodes to exchange messages, making the communication explicit. CQuest specifies that the contention window is composed of a series of slots, and to contend, each node selects a slot and waits to hear from other nodes until its chosen slot arrives, at which point it transmits and wins the contention phase. To solve this problem in the Bluetooth implementation using unicast connections, CQuest was modified so that the scanner for any round i, denoted $scanner_i$, becomes responsible for running the contention phase for round $i + 1$. First, at the beginning of round i, $scanner_i$ performs Bluetooth device discovery to find all nodes in its Bluetooth neighborhood that run the CQuest protocol. Then, at the beginning of the contention phase, each node selects a slot value for itself as described before. However, no contention packet gets sent based on the chosen value. Instead, $scanner_i$, based on the list of Bluetooth neighbors it has acquired earlier in the round, connects directly to each member of that list one by one and asks each neighbor what value it has chosen. Then, after querying all neighbors, $scanner_i$ determines who the scanner will be for round $i + 1$ (in case of a tie, it chooses randomly) and contacts that node directly to inform it that it should perform scanning for the next round. If the Bluetooth cluster is multi-hop, it becomes all the more difficult to execute the contention phase successfully.

CQuest had to be further modified to deal with the scanner leaving its cluster. This is because, unlike the original protocol, a contention phase over Bluetooth cannot take place in the absence of a scanner. To handle this issue, each node keeps a timer based on the expected interval between contention phases, and the time by which it can expect to be contacted if there was a scanner nearby. When that timer expires, nodes probabilistically decide to “run” the contention phase themselves after a randomly chosen period, unless any other scanner contacts it before then.

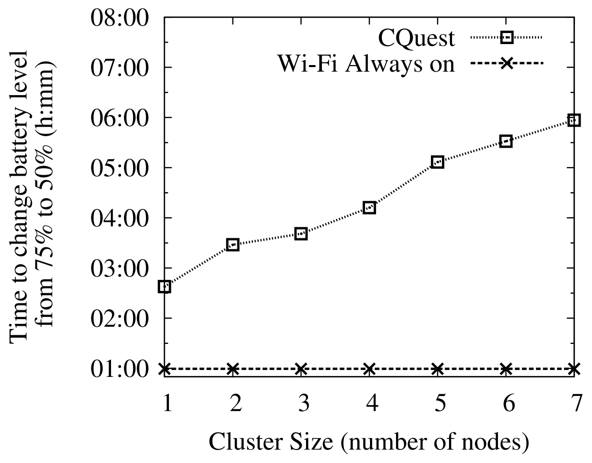


Figure 4.7: Effect of Cluster Size on Energy

4.4.2 Evaluation

Since our main goal is energy-efficiency, we evaluated whether CQuest can enable the phones to coordinate and use less energy. For this experiment, we used a static cluster and varied the size of the cluster size from 1 to 7. The cluster was a clique (i.e., all phones were in Bluetooth range of each other). If the nodes can coordinate successfully, for an n-node clique, each node should spend only $\frac{1}{n}$ of the energy it spends when it scans by itself. For measuring energy savings, we looked at how long it took on average for the battery levels of the phones to decrease from 75% to 50%. The protocol used for Wi-Fi discovery was U-Connect with $p = 7$. In our experiments, the phones were able to successfully coordinate and the average time to go from 75% to 50% grew almost linearly with an increase in cluster size. Also, as a base case comparison, we show the time it takes for a similar change in battery level when the Wi-Fi interface remains always ON.

4.5 Conclusions and Future Directions

The contribution of this work is a new coordinated neighbor discovery protocol that addresses the problem of finding peers in opportunistic networks by leveraging the clustering of nodes and the presence of radio heterogeneity. Extensive simulation results show that CQuest discovers almost the same number of contacts as a regular Wi-Fi discovery protocol, while reducing the energy cost drastically. Interestingly, the loss of these contacts had little or no impact on a common DTN routing protocol.

Current design of CQuest requires a node to collaborate with nearby nodes whenever possible. However, when nodes moving at different speeds collaborate (e.g., one static and one walking), it can result in lost discovery opportunities. It would be interesting to investigate whether the collaboration should be restricted only to static nodes or mobile nodes that are moving together.

In simulation, CQuest needs to be evaluated with additional mobility models and against other approaches to dual-radio communication. For example, instead of actively searching for neighbors directly through the long-range radio, one possible alternative is to use it only to retain connectivity as needed. Specifically, when a neighbor gets discovered with the short-range radio but then moves away from the range of that radio, the long-range radio can be used to maintain connection with that neighbor. This approach will not work in a dense network since a node would always have one or more such connections to maintain and thus the scheme will end up having the long-range radio “on” most of the time for keeping those connections alive. However, in a sparse network, this scheme can ensure that the long-range radio gets used only when needed and thus may reduce overall energy cost.

The implementation of CQuest on smartphones was evaluated only on a static setting. Further experiments need to be done to evaluate how the protocol performs with real life mobility and whether the implementation scales well with larger cluster sizes.

It would be useful to explore other applications for CQuest’s approach of cluster-based collaboration. For example, instead of having each node independently determine its own location, nodes within a cluster can collaborate to find out their common location. This can result in substantial energy savings at the cost of reduced accuracy.

Chapter 5

Leveraging Clustering for Opportunistic Routing:*Mercury*

Protocols for opportunistic networks have until this point focused primarily on overcoming the partitioning of those networks. However, emerging studies show that opportunistic networks also exhibit considerable clustering. Neither current store-carry-forward protocols nor path-based protocols are equipped to effectively handle both of these network characteristics. In response, we present the design of lightweight clustering techniques that allow our protocol, Mercury, to perform well in both partitioned and clustered environments by integrating store-carry-forward techniques with path-based techniques when appropriate. Our evaluations show that Mercury can significantly increase delivery ratio (from 13% to 80% over existing protocols with real life traces) while lowering average latency considerably and this improvement in both the performance metrics comes at an acceptable control overhead.

5.1 Routing in Opportunistic Networks

Existing routing protocols for opportunistic networks have typically focused on handling only partitioning. The most simple SCF-based opportunistic approach is to implement an epidemic-style protocol that attempts to replicate messages during every node encounter [64]. More complex flooding-based schemes have been proposed that learn and utilize information from the network to achieve better performance (e.g., MaxProp [18], RAPID [13], and PRoPHET [35]). However, when resources like buffer size and energy are limited, the cost of flooding-based schemes becomes prohibitive. *Quota-based* forwarding protocols, such as Spray and Wait [61], and EBR [43], try to address this by limiting the number of times a message can be replicated.

Although such SCF protocols work well in highly partitioned mobile networks by focusing only on one-hop encounters, they become extremely inefficient when networks are more connected and end-to-end paths are often available, as in mobile ad hoc networks (MANETs). Most existing protocols designed for MANETs try to discover a path to the destination - either pro-actively (e.g., DSDV [51], OSLR [16]) or reactively (e.g., AODV [50], DSR [28]) - before the data is actually sent. Other protocols use opportunistic means (e.g., ExOR [15]) or geographical knowledge (e.g., GPSR [30]) to progressively forward the data towards the destination without prior path discovery. However, when networks get partitioned and routes to destinations become unavailable, all of these protocols give up after only a few routing failures and discard the messages.

Such a failure model has led to path-based approaches being completely discarded in favor of SCF schemes when dealing with intermittent connectivity. However, the existence of clusters in such fragmented networks has opened up the potential to devise a new approach that can employ *cluster-aware* mechanisms to leverage SCF and path-based techniques and efficiently utilize message delivery opportunities both within and across clusters.

5.1.1 Routing in Clustered Partitioned Networks

To enable routing in networks that exhibit both clustering and partitioning, previous work [45, 40, 21] has proposed using path-based routing for intra-cluster delivery and falling back to SCF approaches when the destination is not part of the same connected component. Chen et al. proposed controlled mobility (i.e., data mules) to bridge stationary clusters and DSDV for intra-cluster traffic [21]. The hybrid scheme in [45] uses AODV to search for suitable SCF-capable nodes during route discovery so that messages can be forwarded to those nodes if no route is found. This approach does not let any node other than the source use path-based routing even though a relay and destination may become part of the same cluster at some later point in time. The Mobile Relay Protocol(MRP) [40] proposed by Nain et al. tries to address this limitation by letting all nodes run a pro-active protocol like DSDV. In this scheme, the message gets disseminated to its immediate neighbors and then this one-hop relay nodes use DSDV to find a connected path to the destination when one exists. Inter-cluster communication in MRP is limited to only one-hop relays and cannot find or utilize path-based routing opportunities arising from cluster encounters. Also, a proactive protocol like DSDV is inefficient when partitioning is very frequent. In fact, the target environment for that scheme was more connected networks like MANETs which led the authors to compare the performance of MRP against only regular DSDV.

Exploiting delivery opportunities during cluster encounters is more challenging since it requires some level of cluster management. Thomas et al. proposed Group-based routing [62] where explicit clustering techniques are augmented with routing at the group level to route messages to the destination *group*, within which the message gets delivered to the destination *node* using path-based routing. However, such an approach works only when groups are very stable and nodes only occasionally relocate from one group to another, making the concept of “destination group” feasible.

Island Hopping [56] assumes the existence of *stable* concentration points (CP) in the network where the average cluster size is above a certain threshold (15 for the traces analyzed in the paper). The flow of nodes between CPs is also assumed to be stable over time. Inter-cluster routing is performed by learning the CP graph gradually and leveraging that knowledge for making forwarding decisions. Hence, the protocol is unsuitable for scenarios where cluster formation is ad hoc and node movement pattern is not very regular.

While some of these existing approaches take a step in the right direction by leveraging clustering, they are limited in their applicability to very specific scenarios and are therefore unable to exploit all delivery opportunities, specifically intra- and inter-cluster communication. Also, all of the combined approaches are tied to a very specific SCF technique. In response, we next present Mercury, our hybrid opportunistic protocol that goes beyond these limitations and effectively integrates path-based routing with *any* of the compatible SCF techniques within a cluster-aware framework to successfully exploit the clustering phenomenon even in very generic opportunistic networks.

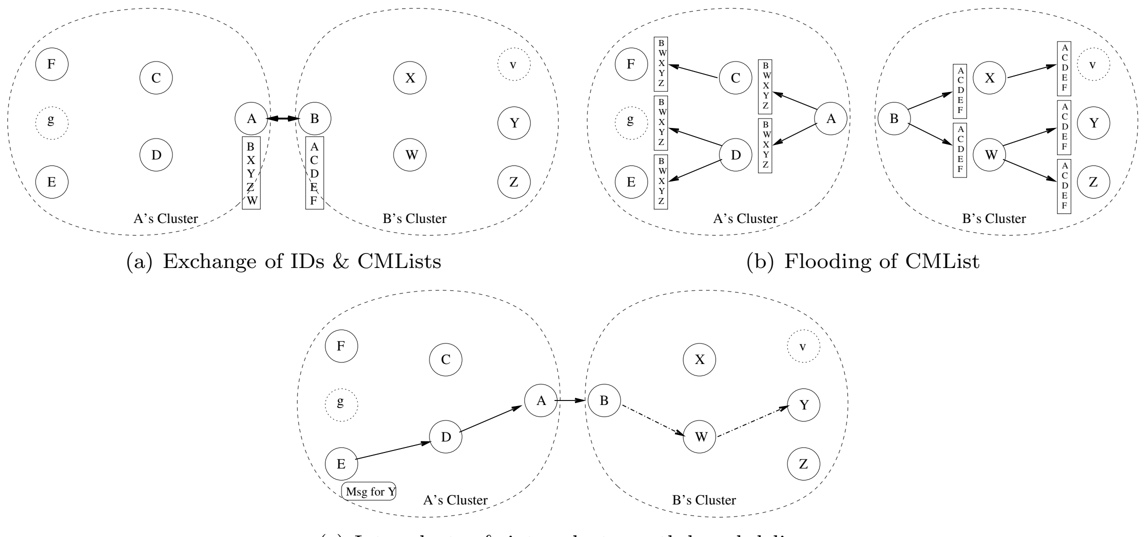
5.2 Mercury

The primary goal of Mercury is to efficiently utilize the formation of transient clusters in opportunistic networks. This is achieved by maintaining *just enough* information about the clusters to help nodes discover multi-hop connectivity when two clusters meet, and then use path-based routing for message delivery. Intra-cluster path-based routing is also supported through reactive route discovery and aggressive caching of routes. Given that the target environment is inherently partitioned, Mercury uses SCF routing as the base mechanism for message delivery, and puts cluster maintenance and path-based routing on top of it to significantly boost the delivery ratio and decrease latency at the cost of reasonably low control overhead.

5.2.1 The Basic Approach

The key mechanism in Mercury is light-weight clustering that facilitates the seamless integration of SCF routing and path-based routing. To support clustering, each node maintains a list of other nodes it believes to be members of its cluster. This is called the Cluster Membership List (CMList). If this list is empty, the node is a single node cluster.

When there is an encounter between two nodes running the Mercury protocol, the nodes first exchange node IDs and CMLists (see Figure 5.1(a)). Since Mercury does not require or enforce consistent views of the cluster between members, these lists may be incomplete. In the example (see Figure 5.1(a)), g is one such node that is part of *A's cluster*, but unknown to A and hence is not included in the CM-



(a) Exchange of IDs & CMLists

(b) Flooding of CMList

(c) Inter-cluster & intra-cluster path-based delivery

Figure 5.1: The Mercury Protocol

List that A gives to B. Although this may result in some lost opportunities for path-based routing across clusters, these messages will likely find delivery opportunities using Mercury's other routing mechanisms, including the base SCF scheme.

After exchanging the CMLists, each node floods the received CMList into their own cluster (see Figure 5.1(b)) to enable all nodes in one cluster to learn about nodes present in the other cluster. To avoid propagating redundant CMLists, nodes compare the exchanged CMList with the last received CMList and floods only if the received CMList is different.

The two nodes then move to the opportunistic forwarding phase and follow the replication strategy of the SCF protocol component of Mercury. If no clustering can be leveraged to improve performance, this step ensures that Mercury's performance is bounded by the underlying SCF protocol. During the direct exchanges, nodes in both clusters are learning about the membership of the other cluster through the distribution of the CMLists. Any node that has a message destined for a node in the other cluster uses path-based routing to deliver the message to the initial node from the other cluster (i.e., A or B). For example, in Figure 5.1(c), node E in A's cluster receives the CMList of B's cluster flooded from A. E then goes through all of the messages in its buffer to see if there are any messages destined for nodes in B's cluster. In this example, node E finds message M that is destined for node Z. Using path-based routing, E first sends message M to B, and then B delivers it to Z, also using path-based routing (see Figure 5.1(c)).

5.2.2 Cluster Maintenance

Given the dynamic nature of the network, Mercury supports clustering by identifying links that are long-lived enough to support multi-hop path-based routing. Only nodes connected through such *stable* links are considered part of a cluster. This notion of cluster membership allows Mercury to limit the propagation of control messages within a group of well-connected nodes and thus filter out single-hop contacts that are too brief to support path-based routing. There are two components of cluster maintenance in Mercury: identifying stable neighbors and building the cluster membership list.

- Identification of stable neighbors: Neighbor discovery in opportunistic networks is generally supported by beacons or simple keep-alive messages. Mercury takes advantage of these existing neighbor discovery mechanisms for evaluating the stability of a connection to a given neighbor. Each node broadcasts `hello` messages containing a sequence number, incremented for each new message. If a node hears $threshold_{join}$ sequential `hello` messages from a neighbor, it marks that neighbor as stable. If $threshold_{leave}$ period passes without receiving a `hello` message from a neighbor, the neighbor is completely removed from the neighbor table. The identification of these one-hop *stable links* is central to the notion of clusters in Mercury.

- Building the Cluster Membership List (CMList): In Mercury, a cluster is any subset of nodes in the network that forms a stable connected component, including single node clusters. To minimize cluster-wide negotiation and overhead, Mercury does not try to enforce a globally consistent view on the cluster. Rather, each node forms its own view of its cluster and maintains a set of node IDs that it believes to be members of its cluster. This Cluster Membership List (CMList) gets populated in two ways. Any node that becomes a stable neighbor of a node (as described above) is immediately inserted into this list. To identify cluster members that are multiple hops away, each node broadcasts a `cluster_hello` message every *cluster_advertisement_period*. Relaying nodes attach their IDs to the message and forward it over all *stable* links. Any node receiving a `cluster_hello` message immediately puts the original sender of the message in their CMList. In a similar manner, since other control messages are also transmitted only over stable links and contain the list of previous hops, Mercury snoops on those messages and updates the CMList as necessary. It is important to note that the appending of node IDs to every message can be expensive in large networks. However, these techniques are only used within clusters, which are expected to be small in size, and so will not incur as much overhead as if used network-wide.

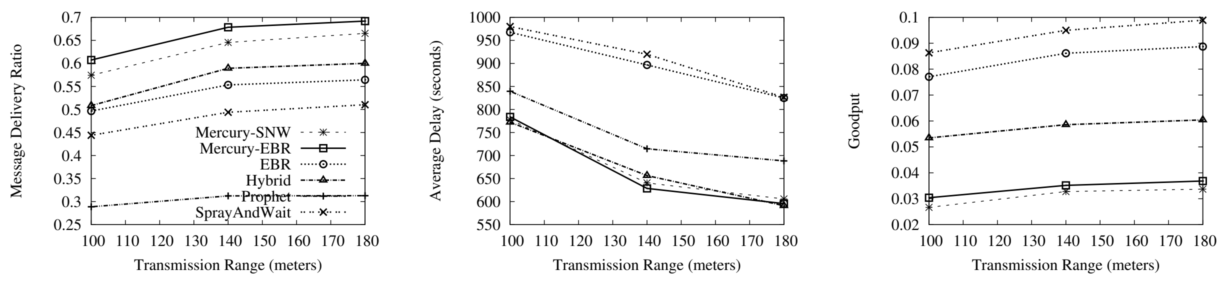
  While stable neighbors are always part of the CMList, destabilization of a stable neighbor does not immediately lead to its removal from the list. The reason is that the node does not know if the single-hop link, which has now become unstable, was the only *stable* path to that node. A node is removed from the CMList only if more than *cluster_advertisement_period* period of time passes after its latest insertion in the set (using either `cluster_hello` or any cluster-wide broadcast message).

5.2.3 Routing

Mercury is designed for networks where partitioning is expected and so the base routing mechanism is SCF. Essentially, the SCF component determines the rules for replication. Unless there is a multi-hop path available to the destination, any data message exchange between two nodes occurs as a result of the execution of the SCF routing component. From a design perspective, Mercury is agnostic to the choice of the exact SCF technique. However, adopting any of the flooding-based approaches [64, 18, 13, 35] would result in a message to be eventually replicated to all members of a connected component, nullifying the performance gain that results from Mercury's use of path-based routing within or across a cluster. On the other hand, quota-based protocols like Spray and Wait [61] and EBR [43] work well as the SCF component because they replicate messages to only a subset of nodes in a cluster, leaving opportunities for Mercury to employ path-based routing and improve performance whenever possible.

Through the use of its clustering mechanisms, Mercury obtains information about paths to destinations within a cluster or across two briefly connected clusters. Mercury then uses path-based routing techniques similar to DSR [28]. However, unlike DSR and many other MANET protocols, if a route fails when a message is in transit, Mercury does not drop the message. Instead, Mercury keeps the message for potential delivery using SCF routing or even future path-based routing in the case that a path becomes available later. By falling back on SCF routing, broken routes do not cause any additional control overhead. However, the message still has later opportunities for delivery. Additionally, by relying on the dynamics of the network to present future path-based routing opportunities, the impact of any undiscovered path-based routing opportunities or imperfect clustering is limited to affecting delay and does not affect successful delivery.

Whenever a node generates a new message, it initiates an attempt to discover a route to the destination by flooding a `RREQ` message within its cluster. Relaying nodes attach their IDs to the message and propagate the broadcast. If the destination is part of the same cluster at that point in time, it receives the message and unicasts a `RREP` back to the source using the route found in the `RREQ`. The source node then unicasts the data message to the destination also using source routing. If the source node does not receive a reply in a sufficient amount of time, it assumes that the destination is not part of its cluster.



(a) Delivery Ratio (b) Average Latency (c) Goodput

Figure 5.2: Cab: varying range

The node stores the message in its buffer for later use and falls back to the SCF mechanism.

To reduce the overhead of issuing a `RREQ` message after the creation of every new message, Mercury aggressively uses a *Route Cache*. To facilitate building up this cache, *all* messages, both control and data, are expected to contain the list of nodes traversed. Therefore, whenever a node receives any message, it can use the path contained in that message to build up routes to all of the nodes that the message has traversed. Upon insertion of a new route into the cache, Mercury goes through all of the data messages in the node's buffer to check if there are any messages for which the new route can be used. This enables Mercury to utilize routes that get created after message generation.

To prevent stale or unstable route information from being inserted into the cache, only messages recently created and received through stable links are considered. To ensure freshness, routes are timed out from the cache every *route_cache_lifetime* period.

5.3 Evaluation

The primary goal of our evaluation is to show that Mercury achieves significant performance gains over other opportunistic protocols by exploiting multi-hop connectivity. To evaluate Mercury's performance, we look at three common performance metrics. *Delivery ratio*, the percentage of total generated messages that eventually get delivered, is the primary measure of performance for opportunistic protocols. *Delay*, our second metric, is defined as the *average latency* per *delivered* message. To evaluate the cost of these two performance metrics, the final metric is *goodput*, defined as the ratio of the number of *delivered* data bytes to the number of total bytes *relayed*, including control messages and dropped messages.

5.3.1 Mobility Models

Since there is no agreed upon mobility model for opportunistic networks, we evaluate the performance of Mercury against other proposed routing protocols in two environments: a real-life cab trace and a random network. The Random Waypoint (RWP) model represents a network where there is nothing inherent in the movement pattern of the nodes to induce clustering. The speed of the nodes were chosen randomly between 5 m/s and 25 m/s and the wait time was uniformly chosen between 60 and 120 seconds.

To evaluate the effect of clustering in a more real-life scenario, we also evaluated Mercury using real life cab traces used in previous work [54, 55]. The data set contains mobility traces of approximately 500 taxi cabs collected over 30 days in the San Francisco Bay area. For our evaluation purposes, we used traces collected between 4 PM to 5 PM, the beginning of afternoon rush hour traffic. To ensure statistically reliable results, we used data collected for ten 10 consecutive days starting from 21st April 2008.

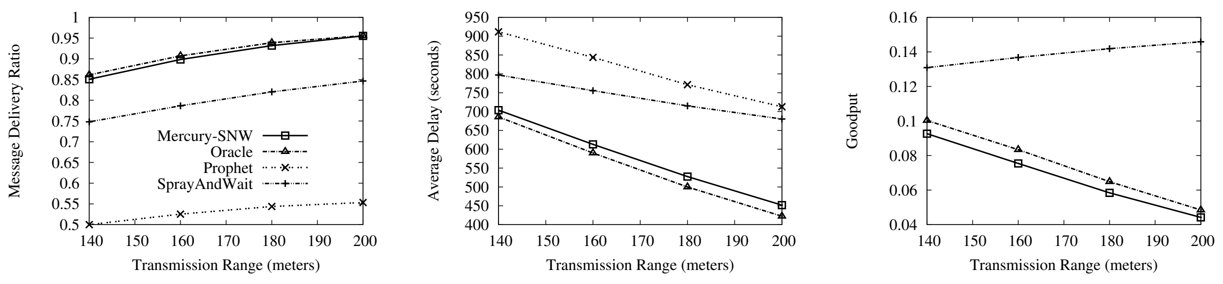
5.3.2 Performance Results

We performed two groups of simulations on each of the two mobility models. First, the transmission range of each node was varied to affect density. For RWP, the range was varied from $140m$ to $200m$. Since the cab traces reflect the mobility of a vehicular network where the actual transmission range is typically smaller, we vary the range from $100m$ to $180m$. Second, the buffer size was varied in each node from 1 MB to 7 MB.

For our simulations, we used the Opportunistic Network Environment (ONE) simulator [31]. For RWP, the area was kept constant at $5600m \times 4000m$. For all runs, the packet size was kept constant at 20 KB and the total number of nodes in all simulations was 250. Each simulation lasted for one simulated hour. Each data point is the average of 10 runs.

For comparison, we evaluated Mercury against a managed-flooding protocol (Prophet) and two quota-based opportunistic protocols (EBR and Spray and Wait). For the trace-based evaluation, We implemented both EBR and Spray and Wait as the SCF component of Mercury and refer to the two implementations as Mercury-SNW and Mercury-EBR respectively. Any performance improvement that Mercury-SNW achieves over Spray and Wait and Mercury-EBR achieves of EBR is clearly the result of leveraging clustering and utilizing path-based routing opportunities. Additionally, we evaluate Mercury against an Oracle, which is similar to Mercury but has perfect knowledge about paths between any two pairs of nodes and does not use or need any cluster-based mechanisms or aggressive snooping to build its routing table. Oracle represents a protocol that uses a very explicit cluster management mechanism since use of such mechanisms would provide the nodes with almost perfect knowledge about connectivity to other nodes.

To better gauge the effectiveness of light-weight clustering, we also evaluated the performance of Mercury without any cluster maintenance. This stripped down version of Mercury, which is referred as



(a) Delivery Ratio (b) Average Latency (c) Goodput

Figure 5.3: RWP: varying range

Hybrid, neither broadcasts any `cluster_hello` messages nor maintains or exchanges any lists of cluster members. However, in addition to SCF using Spray and Wait, it does perform other Mercury functions like route discovery, aggressive snooping and route caching. In other words, Hybrid represents a cluster-unaware simple combination of SCF and path-based routing.

Due to space constraints and for the sake of clarity, we divided the protocols into two sets (not mutually exclusive) and ran one set of protocols for each scenario. For the cab traces, we report results for Mercury-SNW, Mercury-EBR, EBR, Hybrid, Prophet and Spray and Wait. For the RWP scenario, we present results for Mercury-SNW, Spray and Wait, Prophet and Oracle. Because of the extremely low goodput of Prophet, we do not show the value for this protocol in any of the goodput graphs.

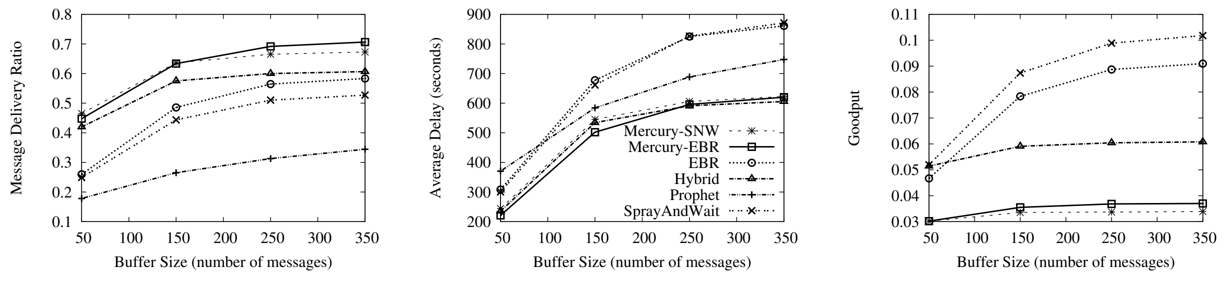
Unless otherwise specified, transmission range is 180 m, traffic load is 1 message per node per minute and buffer size is 5 MB (enough to hold 250 20KB messages). TTL of all generated messages are set to a large enough value so that they do not expire.

- Varying Density: We start by evaluating the effect of density by varying the transmission range. Since a higher transmission range means more contacts between nodes, the delivery ratio for all protocols should improve. Additionally, higher density means more cluster formation, which should benefit Mercury even more.

  In the cab scenario, the delivery ratio for both versions of Mercury improves sharply with an increase in transmission range from 100m to 140m, but then the rate of improvement starts diminishing (see Figure 5.2(a)). This is because the number and size of clusters increase initially with the increase in transmission range. However, since the nodes are not spread uniformly across the area, increasing transmission range does not increase the cluster sizes after a certain value. At best, some multi-hop connectivity becomes single hop direct connections, which does not provide any benefit for Mercury since it has already exploited the available multi-hop paths. For EBR, Prophet and Spray and Wait, the increase is comparatively small, yet steady, since these protocols only leverage the increase in direct contacts resulting from higher transmission ranges. Hybrid delivers more messages than both EBR and Spray And Wait. This shows that augmenting SCF protocols with even simple route discovery mechanisms and aggressive snooping yields performance benefits over the base SCF protocol. For average latency, the improvement for the two versions of Mercury is again much steeper than it is for the other protocols, since more messages get delivered quickly due to the increased clustering (see Figure 5.2(b)). On the other hand, both versions of Mercury-SNW perform much better than Hybrid in terms of delivery ratio. This gap in performance is solely due to Mercury's clustering mechanisms that Hybrid lacks. This suggests that simply using a path-based protocol side by side an SCF protocol is not sufficient to fully leverage clustering in disconnected networks. In the goodput metric, Hybrid outperforms Mercury-SNW because it does not incur the overhead of cluster maintenance. On average, Mercury achieves around 22% improvement in delivery ratio and 23% decrease in latency over all transmission ranges in comparison to the best performing SCF protocol. On the other hand, both versions of Mercury lag behind Spray and Wait and EBR in the goodput (see Figure 5.2(c)). This can be considered as an acceptable cost for significant gain in *both* delivery ratio and latency.

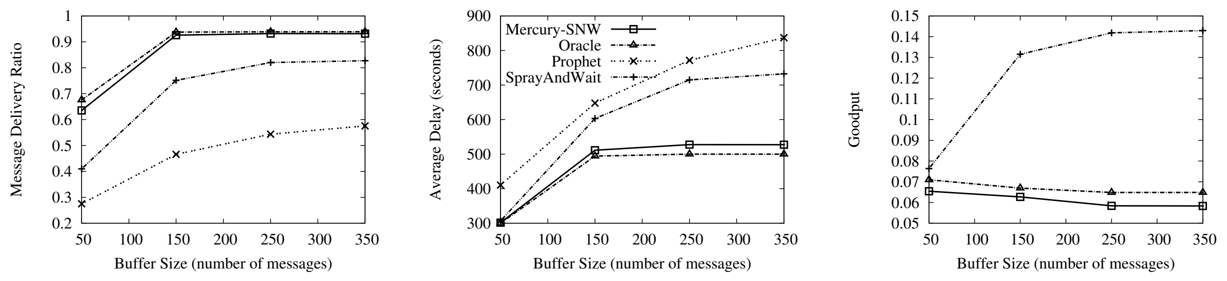
For RWP, an increase in transmission range increases the delivery ratio similarly for all four protocols (see Figure 5.3(a)). Mercury-SNW maintains its edge over the other protocols irrespective of transmission range and delivers more than 95% of the messages when the transmission range gets to $200m$. The average latency of all messages drops with an increase in transmission range (see Figure 5.3(b)). However, the drop for Mercury-SNW is much sharper. This is because a denser network creates more routing opportunities for Mercury-SNW and more messages get delivered quickly using opportunistic routing. For $200m$, the decrease in latency in comparison to Spray and Wait becomes as high as 33%. The difference in goodput between Mercury-SNW and Spray and Wait starts widening with an increase in transmission range (see Figure 5.3(c)). This is because the control overhead for Mercury-SNW due to cluster management messages increases in a denser network. Oracle is expected to provide an upper bound on the performance because of its perfect knowledge. In the RWP scenario, Oracle does perform better than Mercury-SNW in terms of both delivery ratio and delay, but the difference is extremely small (less than .01% for delivery ratio and around 5% for latency). In terms of latency, Oracle performs best since whenever there is a path available, it gets to know about it instantaneously. Goodput of Oracle is slightly higher than Mercury-SNW. This narrow difference suggests that the cost of control overhead is not significant for Mercury.

- Varying Buffer Size: Since most opportunistic protocols replicate messages, buffer size is one of



(a) Delivery Ratio (b) Average Latency (c) Goodput

Figure 5.4: Cab: varying buffer size



(a) Delivery Ratio (b) Average Latency (c) Goodput

Figure 5.5: RWP: varying buffer size

the major performance constraints. With a medium to heavy load, all protocols should benefit from increased buffer capacity. On the other hand, average latency can be expected to increase since nodes drop fewer messages and so retain messages longer in their buffers, increasing the likelihood of successful delivery at a later time. Varying buffer sizes should not affect Mercury's cluster leveraging techniques. Indeed, the results confirm this intuition. The relative performance gain of Mercury over the base SCF protocols remains constant over different buffer sizes. These results also confirm Mercury's resource friendliness since it achieves the highest delivery ratio of all protocols.

In the cab scenario, both versions of Mercury performs best for all buffer sizes (see Figure 5.4(a)). Even when the buffer size is set to 1 MB (50 messages), and the delivery ratios of all other SCF protocols go below 25%, both versions of Mercury-SNW achieves a delivery ratio of around 45% which is more than 80% improvement over the nearest performing protocol. As expected, the average latency of delivered messages for all protocols increases with buffer size. (see Figure 5.4(b)). The two versions of Mercury, for all buffer sizes, maintains the lowest average latency which is on average around 25% lower than any of other existing protocols.

For goodput (see Figure 5.4(c)), Spray and Wait and EBR do better than Mercury and this gain increases with increase in buffer size. This a result of the buffer constraint being relaxed, which enables SCF protocols to drop fewer messages and so perform better, boosting the goodput value. Mercury-SNW and Mercury-EBR benefit comparatively less from buffer size increase since performance is not overly constrained by the small buffer space.

In the RWP model, Mercury-SNW again achieves the highest delivery ratio for all buffer sizes (see Figure 5.5(a)) and achieves more than 50% improvement for the lowest buffer size. However, it is interesting to note that the delivery ratio for Mercury-SNW does not improve much for buffer sizes larger than 3 MB. This is because Mercury-SNW achieves an extremely high delivery ratio even for small buffers and does not have much room for improvement when this constraint is relaxed. While both Prophet and Spray And Wait always benefit from larger buffers, the rate of gain starts diminishing after 3 MB. Oracle performs better than Mercury-SNW for smallest buffer size, but this difference disappears after a certain threshold as performance of both protocols flattens out.

Similarly, average latency for messages delivered by Mercury-SNW reaches a constant value at 3 MB (150 messages), which is always lower than that achieved by any of the other protocols (see Figure 5.5(b)). In terms of goodput, Mercury-SNW is only slightly worse than Spray And Wait (see Figure 5.5(c)) and much better than Prophet.

The above results show that at certain buffer sizes, the rate of gain begins to taper off. This suggests that there is a threshold buffer size in this particular setting, beyond which the buffer size does not remain a big constraint. This is the reason we limited our evaluation to 7 MB although available memory on most devices can be expected to be bigger.

5.3.3 Evaluation of Mercury parameters

In the design of Mercury, we introduced several protocol parameters, *cluster_advertisement_period*, *route_cache_lifetime* and $threshold_{join}$, which are essential to the efficient operation of Mercury. In our simulations, we empirically determined the best setting for each of these parameters. However, our evaluations showed that for the first two parameters, there is only a small penalty (less than 1% across all values) for incorrectly setting the parameter. The setting for $threshold_{join}$ had a larger influence on metric performance. This parameter expresses a trade-off of higher message overhead for more accurate cluster knowledge. Our evaluations showed a linear decrease in delivery rate and delay, but a linear increase in goodput as the $threshold_{join}$ period increases.

5.4 Conclusions and Future Directions

Clustering often presents multi-hop routing opportunities that the current opportunistic protocols are incapable of taking advantage of. In comparison our protocol, Mercury, takes advantage of this clustering by intelligently integrating store-carry-forwarding and path-based routing on top of a light-weight clustering substrate. Our evaluation of Mercury shows that the benefits from using the additional path-based routing can be significant. In two different scenarios and across three different parameters, Mercury outperforms other well-known cluster-agnostic opportunistic protocols convincingly both in terms of delivery ratio and delay, without incurring too high overhead.

In the future, Mercury can be extended to dynamically learn about different cluster dynamics and adapt its forwarding policies accordingly. Specifically, in scenarios where nodes tend to move in groups, Mercury should be able to treat contacts between different clusters as forwarding opportunities. This would increase the amount of data handled by gateway nodes during inter-cluster contacts and might require message aggregation. Knowledge of long term stability of clusters will also open up the possibility for Mercury to perform intra-cluster load management as well as optimize the path-based route management component.

Chapter 6

Conclusions and Future Directions

Opportunistic networks are critical to supporting communication in environments where infrastructure is unavailable. As mobile wireless devices become more ubiquitous, we expect more applications to emerge that will utilize the potential of direct peer-to-peer communication. In this proposal, we have identified and contributed to components that will help make such opportunistic communication more efficient and thereby, more feasible. In particular, we have shown that by taking advantage of the clustering phenomenon inherently found in these networks, nodes can cooperate to efficiently detect and exploit contact opportunities and improve performance of opportunistic protocols significantly.

All of the different components of this work have individually opened up new avenues for further research that we have already discussed. Additionally, the whole work can be greatly enhanced along two directions. First, a more detailed understanding of the clustering phenomenon would enable the cluster-aware protocols to perform better. Second, detection and utilization of contact opportunities need to be integrated into a single unified protocol.

6.1 Necessity of a more detailed Understanding of Cluster Dynamics

While previous approaches have established the existence of clustering as an integral characteristic of mobile partitioned networks, and have analyzed the dynamics of clustering to some extent, there still remain properties of this phenomenon that have not been analyzed but might be beneficial for both discovery and routing in opportunistic networks.

6.1.1 Cluster Diameter

Diameter of a cluster is defined as the maximum length of the shortest *connected* path between any two members within the same cluster. It is important to know the diameter of clusters that get formed in real life because even though a cluster may be large, most of its nodes may be directly connected to each other, rendering multi-hop routing techniques unnecessary.

6.1.2 Inter-cluster Encounters

While contacts between individual nodes have received a lot of attention, encounters between clusters have not yet been investigated. This is all the more interesting for vehicular networks where road constraints often force nodes to move in groups. Another interesting thing to look at will be the distribution of node association time with clusters, instead of just looking at the median [54]. For example, it might happen that most nodes remain in the cluster for a reasonable time, and only a small fraction of the nodes have really small association times. In that case, those small association times need to be treated separately than merge events. Those would be cases of one node brushing past a cluster. Using those brief encounters efficiently would require some cluster nodes to act as temporary gateway nodes.

6.1.3 Effect of Radio Heterogeneity

While the effect of different radio transmission ranges on the number of total clusters have been analyzed before, the effect of radio heterogeneity has not been studied. This becomes important when devices are equipped with radios that have significantly different ranges, e.g., WiFi and Bluetooth. While such analysis will not be difficult with vehicular traces that have actual GPS coordinates of the nodes, it will be difficult to figure out the effect of long-range radio on human networks since most traces based on human mobility have information only about contacts through the low-power radio, not the actual position of the nodes.

6.2 Integration of Discovery and Message Routing

Efficient discovery and message forwarding, are both important components of opportunistic communication and any practical deployment should contain the ability to perform both functions. However, there is currently no work on how discovery and transmission of data message interact and affect each another in an opportunistic environment, especially when energy is a scarce resource. This is a unique challenge for the contact-driven communication paradigm since discovery needs to be a continuous process. It is desirable to have the two components work cooperatively. The main challenge arises from the possibility that data transmission between discovered nodes might interfere with discovery beacons and increase the latency to find new neighbors. Also, when nodes are equipped with multiple radios and take a coordinated approach to neighbor discovery, forwarding data to the intended node might require taking into consideration the energy-per-bit characteristics of radios and other energy costs. A high-bandwidth radio like IEEE 802.11 consumes more power but significantly reduces the time for transmissions. Consequently, it offers net savings in total communication energy when there is enough data to offset wake-up energy overhead.

References

[1] Aka-aki. http://akakicom.

[2] Bluehoo. http://bluehoo.com.

[3] Jabberwocky. http://www.urban-atmospheres.net/Jabberwocky/info.htm.

[4] Lokast. http://www.nearverse.com/lokast.

[5] Maemo - open source operating system. http://maemo.org.

[6] ns2 network simulator. http://www.isi.edu/nsnam/ns/.

[7] A. R. Abhyankar, S. A. Soman, and S. A. Khaparde. Min-max fairness criteria for transmission fixed cost allocation. *IEEE Transactions on Power Systems*, 22:2094–2104, 2007.

[8] Y. Agarwal, T. Pering, R. Want, and R. Gupta. Switchr: Reducing system power consumption in a multi-client, multi-radio environment. In *Wearable Computers, 2008. ISWC 2008. 12th IEEE International Symposium on*, pages 99–102, Oct 2008.

[9] Yuvraj Agarwal, Ranveer Chandra, Alec Wolman, Paramvir Bahl, Kevin Chin, and Rajesh Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, MobiSys '07, pages 179–191, New York, NY, USA, 2007. ACM.

[10] Yuvraj Agarwal, Curt Schurgers, and Rajesh Gupta. Dynamic power management using on demand paging for networked embedded systems. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, ASP-DAC '05, pages 755–759, New York, NY, USA, 2005. ACM.

[11] Ganesh Ananthanarayanan and Ion Stoica. Blue-fi: enhancing wi-fi performance using bluetooth signals. In *Mobisys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 249–262, New York, NY, USA, 2009. ACM.

[12] Paramvir Bahl, Atul Adya, Jitendra Padhye, and Alec Walman. Reconsidering wireless systems with multiple radios. *SIGCOMM Comput. Commun. Rev.*, 34:39–46, October 2004.

[13] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. DTN routing as a resource allocation problem. In *Proc. ACM SIGCOMM*, 2007.

[14] Nicola Baldo, Federico Maguolo, Marco Miozzo, Michele Rossi, and Michele Zorzi. ns2-miracle: a modular framework for multi-technology and cross-layer support in network simulator 2. In *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, Value-Tools '07, pages 16:1–16:8, ICST, Brussels, Belgium, Belgium, 2007. ICST.

[15] Sanjit Biswas and Robert Morris. Exor: opportunistic multi-hop routing for wireless networks. *SIGCOMM Comput. Commun. Rev.*, 35(4):133–144, 2005.

[16] Josh Broch, David A. Maltz, David B. Johnson, Yih C. Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. 4th ACM/IEEE MobiCom*, 1998.

[17] J. Burgess, Br. Gallagher, D. Jensen, and B. Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. In *Proc. IEEE INFOCOM*, 2006.

[18] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. In *Proc. IEEE INFOCOM*, April 2006.

[19] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: an approach to interplanetary internet. *Communications Magazine, IEEE*, 41(6):128–136, June 2003.

[20] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007.

[21] Yang Chen, Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. Hybrid routing in clustered dtns with message ferrying. In *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, 2007.

[22] Prabal Dutta and David Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 71–84, New York, NY, USA, 2008. ACM.

[23] Prabal Dutta and David Culler. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 71–84, New York, NY, USA, 2008. ACM.

[24] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[25] Simon Heimlicher, Merkourios Karaliopoulos, Hanoch Levy, and Thrasyvoulos Spyropoulos. On leveraging partial paths in partially-connected networks. In *INFOCOM*, pages 55–63, 2009.

[26] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *ACM SigComm workshop WDTN*, 2005.

[27] Wei jen Hsu and Ahmed Helmy. On nodal encounter patterns in wireless lan traces. *IEEE Transactions on Mobile Computing*, 9:1563–1577, November 2010.

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

[29] Arvind Kandhalu, Karthik Lakshmanan, and Ragunathan (Raj) Rajkumar. U-connect: a low-latency energy-efficient asynchronous neighbor discovery protocol. In *IPSN '10: International Conference on Information Processing in Sensor Networks*, pages 350–361, 2010.

[30] B. Karp and HT Kung. GPSR: greedy perimeter stateless routing for wireless networks. *Proc. 6th ACM MobiCom*, 2000.

[31] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proceedings of SIMUTools '09*, 2009.

[32] Su Min Kim, Jo Woon Chong, Byoung Hoon Jung, Min Suk Kang, and Dan Keun Sung. Energy-aware communication module selection through zigbee paging for ubiquitous wearable computers with multiple radio interfaces. In *Wireless Pervasive Computing, 2007. ISWPC '07. 2nd International Symposium on*, 2007.

[33] Shouwen Lai, Binoy Ravindran, and Hyeonjoong Cho. Heterogenous quorum-based wakeup scheduling in wireless sensor networks. *IEEE Transactions on Computers*, 99(PrePrints), 2010.

[34] Juan Li and Samee Ullah Khan. Mobisn: semantics-based mobile ad hoc social network framework. In *Proceedings of the 28th IEEE conference on Global telecommunications*, GLOBECOM'09, pages 883–888, Piscataway, NJ, USA, 2009. IEEE Press.

[35] A. Lindgren, A. Doria, and O. Scheln. Probabilistic routing in intermittently connected networks. In *SIGMOBILE Mobile Computing and Communication Review*, 2004.

[36] M. Matuszewski, N. Beijar, J. Lehtinen, and T. Hyyrylainen. Understanding attitudes towards mobile peer-to-peer content sharing services. In *Portable Information Devices, 2007. PORTABLE07. IEEE International Conference on*, pages 1–5, 25-29 2007.

[37] Michael J. McGlynn and Steven A. Borbash. Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 137–145, New York, NY, USA, 2001. ACM.

[38] Mehul Motani, Vikram Srinivasan, and Pavan S. Nuggehalli. Peoplenet: engineering a wireless virtual social network. In *Proceedings of MobiCom 2005*, pages 243–257, New York, NY, USA, 2005. ACM.

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

[40] Delphine Nain, Noshirwan Petigara, and Hari Balakrishnan. Integrated routing and storage for messaging applications in mobile ad hoc networks. In *Proc. of WiOpt 2003, Sophia Antipolis*, pages 595–604, 2003.

[41] Samuel Nelon. Encounter-based routing in disaster recovery networks. Master's thesis, University of Illinois at Urbana-Champaign, 2008.

[42] S. Nelson, A. Harris, and R. Kravets. Event-driven, role-based mobility in disaster recovery networks. In *Proceedings of ACM CHANTS*, 2007.

[43] Samuel C. Nelson, Mehedi Bakht, and Robin Kravets. Encounter-based routing in dtns. In *Proc. IEEE INFOCOMM*, April 2009.

[44] Ivan Niven and Herbert S. Zuckerman. *An Introduction to the Theory of Numbers.* John Wiley and Sons (WIE), 1991.

[45] Jörg Ott, Dirk Kutscher, and Christoph Dwertmann. Integrating DTN and MANET routing. In *Proceedings of CHANTS*, pages 221–228, 2006.

[46] Jeongyeup Paek, Joongheon Kim, and Ramesh Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 299–314. ACM, 2010.

[47] J.R. Pedrasa, M.A. Pedrasa, and A.P. Seneviratne. Information exchange for enhanced network selection. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6, 2010.

[48] T. Pering, Y. Agarwal, R. Gupta, and R. Want. Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *MobiSys 2006: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232, New York, NY, USA, June 2006. ACM Press.

[49] Trevor Pering, Vijay Raghunathan, and Roy Want. Exploiting radio hierarchies for power-efficient wireless device discovery and connection setup. In *VLSID '05: Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, pages 774–779, Washington, DC, USA, 2005. IEEE Computer Society.

[50] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of IEEE WMCSA*, February 1999.

[51] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[52] Anna K. Pietiläinen, Earl Oliver, Jason Lebrun, George Varghese, and Christophe Diot. MobiClique: middleware for mobile social networking. In *WOSN '09: Proceedings of the 2nd ACM workshop on Online social networks*, pages 49–54. ACM, August 2009.

[53] Michał Piórkowski. Sampling urban mobility through on-line repositories of gps tracks. In *Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements*, HotPlanet '09, pages 1:1–1:6, New York, NY, USA, 2009. ACM.

[54] Michal Piórkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. On clustering phenomenon in mobile partitioned networks. In *Proceedings of the 1st ACM SIGMOBILE Workshop on Mobility models*, 2008.

[55] Michal Piorkowski, Natasa Sarafijanovoc-Djukic, and Matthias Grossglauser. A Parsimonious Model of Mobile Partitioned Networks with Clustering. In *The First International Conference on COMmunication Systems and NETworkS (COMSNETS)*, January 2009.

[56] N. Sarafijanovic-Djukic, M. Pidrkowski, and M. Grossglauser. Island hopping: Efficient mobility-assisted forwarding in partitioned networks. *IEEE Sensor and Ad Hoc Communications and Networks*, 2006.

[57] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD trace cambridge/haggle/imote/infocom (v. 2006-01-31). http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote/infocom, Jan 2006.

[58] Cigdem Sengul, Mehedi Bakht, Albert F. Harris III, Tarek F. Abdelzaher, and Robin Kravets. Improving energy conservation using bulk transmission over high-power radios in sensor networks. In *ICDCS*, pages 801–808, 2008.

[59] Eugene Shih, Paramvir Bahl, and Michael J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 160–171, New York, NY, USA, 2002. ACM.

[60] Jacob Sorber, Nilanjan Banerjee, Mark D. Corner, and Sami Rollins. Turducken: hierarchical power management for mobile devices. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 261–274, New York, NY, USA, 2005. ACM.

[61] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proc. SIGCOMM Workshop on Delay-tolerant networking*, 2005.

[62] Markose Thomas, Arobinda Gupta, and Srinivasan Keshav. Group based routing in disconnected ad hoc networks. In *HiPC*, pages 399–410, 2006.

[63] Yu-Chee Tseng, Chih-Shun Hsu, and Ten-Yueng Hsieh. Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. In *INFOCOM*, 2002.

[64] Amin Vahdat and David Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, Department of Computer Science, Duke University, April 2000.

[65] Long Vu, Do Quang, and Klara Nahrstedt. Lessons learned from bluetooth/wifi scanning deployment in university campus. Technical report, Department of Computer Science, University of Illinois, 2010.

[66] Long Vu, Ivica Rimac, Volker Hilt, Markus Hofmann, and Klara Nahrstedt. ishare: Exploiting opportunistic ad hoc connections for improving data download of cellular users. In *PerGroup/Globecom*, 2010.

[67] J. Yoo and K. Park. A cooperative clustering protocol for energy saving of mobile devices with wlan and bluetooth interfaces. *Mobile Computing, IEEE Transactions on*, PP(99):1, 2010.

[68] Lan Zhang, Xuan Ding, Zhiguo Wan, Ming Gu, and Xiang-Yang Li. Wiface: a secure geosocial networking system using wifi-based multi-hop manet. In *MCS '10: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services*, pages 1–8. ACM, 2010.

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

[70] Wenrui Zhao and Mostafa H. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, FTDCS '03, pages 308–, Washington, DC, USA, 2003. IEEE Computer Society.

[71] Rong Zheng, Jennifer C. Hou, and Lui Sha. Asynchronous wakeup for ad hoc networks. In *MobiHoc 2003*, pages 35–45, New York, NY, USA, 2003. ACM.

[72] Ruogu Zhou, Yongping Xiong, Guoliang Xing, Limin Sun, and Jian Ma. Zifi: wireless lan discovery via zigbee interference signatures. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, MobiCom '10, pages 49–60, New York, NY, USA, 2010. ACM.