ROUTING PROTOCOLS FOR K-HOP NETWORKS

BY

WILLIAM DOUGLASS LIST

B.S., Rensselaer Polytechnic Institute, 2001

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

Routing Protocols for k-hop Networks

Approved by
Supervising Committee:

_____

_____

ABSTRACT

Previous research has identified the scalability limitations of stand-alone wireless ad hoc networks. Shared access to a single wireless channel in the presence of a dense network leads to congestion and high channel contention. Increasing the diameter of the network introduces long paths that become prone to breakage, leading to degraded performance. As such, in this thesis we propose that the scope of a node's ad hoc capabilities be limited to a maximum diameter of k hops where k is small, in order to trade off connectivity with overhead. In addition, we suggest an overlay of fixed base stations equipped with a secondary high-bandwidth channel to support connections between nodes in the network, with a maximum of $2k$ wireless hops. We present our k-hop network architecture and propose several routing protocols specifically tailored for the architecture. We demonstrate through simulations that these routing protocols provide efficient and reliable routing even in the presence of mobility and high node density. Finally, we describe our testbed here at UIUC that has put into practice the k-hop architecture.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Nitin Vaidya, for his guidance in preparing this thesis. I would also like to thank my family and friends for their support and suggestions. Finally, I would like to thank my research group for their insights and comments. This thesis would not have been possible without the aforementioned individuals.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

MANETs[1] consist of a collection of mobile nodes that act in a distributed fashion without an established infrastructure. Each node in the network serves as a router for forwarding packets on behalf of other nodes, supporting multi-hop communications. MANETs are quickly becoming popular, with many potential applications already identified [1]. However, these wireless networks alone provide limited capacity, making widespread deployment with many participants difficult [2, 3, 4, 5]. Forwarding decisions in a MANET are based on the *routing protocol*, a distributed algorithm that affords each node a partial or complete view of the network topology. A large percentage of existing routing protocols are based on a *flat* architecture, an inherently non-scalable architecture both in terms of control overhead and storage [6, 7, 8, 9]. These protocols work best under a limited range of hops, possessing a so-called *ad hoc horizon* [10]. Some examples of flat routing protocols include but are not limited to DSR [11, 12], AODV [13, 14], TORA [15], SSA [16], and ABR [17] (all reactive), and DSDV [18], WRP [19], STAR [20], GSR [21], FSR [22, 23], HSLS [9] OLSR [24, 25], and TBRPF [26, 27] (all proactive). Reactive routing protocols discover paths on-demand (i.e., only when requested by a node). On the other hand, proactive routing protocols maintain paths to all nodes in the network regardless of whether or not they are being used. A combination of the two techniques forms the basis of routing protocols ZRP [28, 29] and LUNAR [10]. A handful of the

[1]Mobile Ad Hoc Networks

above protocols (FSR, HSLS and ZRP) make use of an *implicit* network hierarchy to aid in scalability, and thus have an advantage in larger networks. This is not to be confused with routing protocols that have an *explicit* hierarchy (see below).

As a result of a lack of adequate scalability both in terms of network capacity and the routing protocols used, recent research has gone in two directions to solve these problems — (1) augmenting the wireless network with an overlay network (utilizing a second channel) [30, 31, 32], and (2) incorporating multiple hierarchies to support aggregation and reduced overhead in the routing protocols (still only one channel) [33, 34, 35, 36, 37, 38, 8, 39, 40, 41, 42, 43]. Both of these methods introduce *explicit* hierarchies in the network. So-called *hybrid networks* that are a combination of the two solutions show great potential and are likely to represent the next generation of modern networking. Thus, there is a strong initiative to develop new architectures and protocols that are customized for these hybrid networks.

The second direction above can take form in ad hoc routing protocols by using the very popular *clustering* paradigm. Some protocols that use cluster forming are the (α, t) Cluster Framework [33], the B-protocol [34], MMWN [35], ZHLS [36], CGSR [37], HSR [38, 8], LANMAR [44, 39], and [40, 41, 42, 43, 45, 46]. These protocols usually benefit from clustering by having smaller routing tables and fewer route updates. However, the clustering approach often implies a high level of complexity in order to create and maintain the clusters. Second, in most clustering schemes each cluster has a single leader (the *clusterhead*) that becomes a traffic hot-spot, especially for inter-cluster communication. Finally, clustering typically leads to non-optimal paths, increasing the delay and traffic of the network.

For reference, good overviews of some of the protocols mentioned thus far can be found in [47, 48].

In this thesis, we introduce a bound on each node's "ad hoc scope", prohibiting it from forming wireless paths that are longer than a parameter k. Small values of k enables us to achieve a high level of connectivity assuming a non-uniform distribution of hosts.

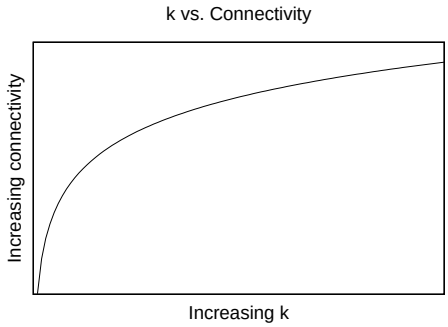

Figure 1.1 Network connectivity versus k.

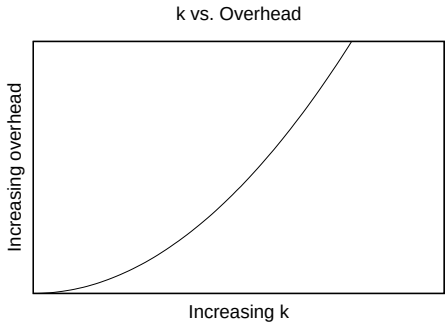

Figure 1.2 Overhead versus k.

As the value of k increases, we predict that the network achieves diminishing returns in terms of the additional connectivity provided, as roughly illustrated in Figure 1.1. At the same time, we predict that the routing overhead shown in Figure 1.2 quickly increases, negating the benefits of the added connectivity. In addition to the k bound, we propose overlaying a fixed network of base stations that take part in the ad hoc network and are fully connected via a secondary high-bandwidth channel. The combination of these two modifications is termed the *k-hop architecture*. This architecture has prompted the development of several routing protocols that take advantage of the new architecture. We describe and compare these different routing protocols in a variety of settings and discuss the trade-offs of each solution. Finally, we provide details on our testbed utilizing the proposed k-hop architecture.

The rest of this thesis is organized as follows. In Chapter 2, we highlight some related work on hybrid networks. Chapter 3 presents the k-hop architecture. We describe our customized routing protocols in Chapter 4. The simulation environment and results of our experiments are shown in Chapter 5. Chapter 6 gives a brief overview of our working testbed. Finally, in Chapter 7, we draw some conclusions and discuss future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

The concept of combining the properties of cellular-based networks and ad hoc networks to create a more versatile system has been the topic of a lot of recent literature. In this section, we discuss some related work.

Early research with hybrid networks involved adding ad hoc capabilities to existing cellular networks [49, 50]. The primary goal of A-GSM [49] is to reduce the occurrence of *dead spots*, or areas in the network where services are not available. This is done by allowing devices to relay through another device to reach a base station. A-GSM uses beacons to select the best device or base station to communicate with. In the *Multihop Cellular Network* [50] (MCN), intra-cell communication between devices can bypass the base station by relaying through other devices, thus potentially improving the spatial reuse of the system.

A significant problem with the cellular and wireless LAN models is *hot spot congestion*, i.e. high network load in a small area. Thus, a collection of approaches have been proposed to reduce the frequency of hot spots, often introducing an ad hoc element to the network [51, 52, 53, 54].

The hybrid approach taken in Sphinx [55] lets flows operate under a time-divisioned channel that is split between a cellular mode and a peer-to-peer (ad hoc) mode. Initially, all flows are served using peer-to-peer mode. When the throughput of a flow degrades

below a threshold (typically due to mobility or increasing number of hops from the base station), it is switched to cellular mode. Sphinx relies on a separate control channel to inform nodes of the current flow schedule. Sphinx was developed based on comparisons done in [56], [57] and [58].

Our k-hop architecture draws similarities to the Zone Routing Protocol (ZRP) [28, 29]. ZRP allows two different routing schemes to be deployed for intra-zone and inter-zone routing. According to [29], ZRP works best with a proactive intra-zone protocol (IARP) [59] and a reactive inter-zone protocol (IERP) [60], as do k-hop networks. Comparatively, our architecture places the burden of inter-zone communication on the network of base stations rather than the ad hoc nodes.

Recent work done by Gerla *et al.* is similar in nature to our k-hop network model [30, 31, 32]. In [32], Extended HSR (based on HSR [38, 8]) is designed to support heterogeneous networks with several routing layers via multiple interfaces. The scheme does not support multi-hop routing, resulting in centralized traffic around the backbone nodes (BN)s. In contrast, our protocol supports multi-hop connections. Furthermore, HSR requires a complicated address translation service, whereas our model has no need for such a service.

The design methodology presented in [30] uses Landmark Ad Hoc Routing (LANMAR) [44, 39] and a clustering technique to build hierarchical networks. An algorithm is employed to select a set of mobile nodes in the network to act as BNs and form the backbone network. DSDV [18] and FSR [22, 23] provide routing at the upper and lower layers, respectively. The k-hop architecture, as will soon see, differs from [30] in several ways. Foremost, a k-hop network makes no use of a clustering algorithm. Nodes in a k-hop network do not form clusters around the base stations. Second, base stations (roughly equivalent to landmarks in LANMAR) in our model rely on the k parameter rather than addresses to define zones. These two differences hold for the next technique described as well.

A new hybrid version of AODV called Hierarchical AODV (H-AODV) is presented in [31]. H-AODV performs RREQs at two separate levels, and improvements in path lengths are

possible by relaying through BNs. H-AODV uses an additional field in the routing control packets to define subnets around each BN. Thus, all packets outside the local subnet must pass through the associated BN. This is not true in our architecture, where nodes are able to locally route to anyone within the k-hop bound. This is especially important for perimeter node communication (see Section 3). A second difference between H-AODV and k-hop networks is that in our architecture nodes are able to utilize one *or more* base stations for routing whereas in H-AODV, each node is associated with only one BN at a time.

CHAPTER 3

THE K-HOP ARCHITECTURE

In this chapter, we describe in detail the properties of our k-hop architecture.

A k-hop network is an ad hoc network comprised of two entities – mobile hosts (MHs) and gateways (GWs). Each MH and GW is equipped with a wireless interface that allows it to communicate with other MHs and GWs over the wireless channel. In addition, each GW possesses a second interface (wired or wireless). These interfaces form a fully-connected backbone network between GWs. If the backbone network consists of wireless links, they are assumed to operate in a different frequency range than the first wireless interface. k in a k-hop network refers to the upper hop bound on all ad hoc connections. For example, in a 2-hop network paths between two MHs or a MH and GW are limited to 2 hops. All nodes reachable within k hops are considered *local*. All other connections are *remote*. By itself, the k bound limits the connectivity of the network, especially if the network spans many hops. To reach a remote MH, sources are required to utilize a GW to "jump" from one part of the network to another. Thus, a guarantee of the k-hop architecture is that every MH can be reached using no more than $2k$ wireless hops.

Figure 3.1 shows a simple 2-hop network with 2 GWs (black circles) and 8 MHs (white circles). Dotted lines indicate connectivity between MHs and GWs. The backbone link between GW **G1** and **G2** is not shown. The following paths in the network are valid.

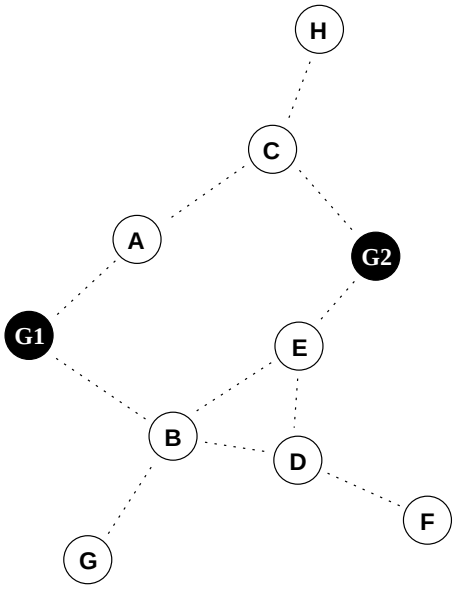

Figure 3.1 A simple 2-hop network.

$\mathbf{A} \Rightarrow \mathbf{D}$[1] could use $\mathbf{A} \rightarrow \mathbf{G1} \rightarrow \mathbf{B} \rightarrow \mathbf{D} \mid 3$[2] or $\mathbf{A} \rightarrow \mathbf{C} \rightarrow \mathbf{G2} \rightarrow \mathbf{E} \rightarrow \mathbf{D} \mid 4$. $\mathbf{E} \Rightarrow \mathbf{C}$ could use $\mathbf{E} \rightarrow \mathbf{G2} \rightarrow \mathbf{C} \mid 2$ or even $\mathbf{E} \rightarrow \mathbf{B} \rightarrow \mathbf{G1} \rightarrow \mathbf{A} \rightarrow \mathbf{C} \mid 4$ if necessary. Third, $\mathbf{G} \Rightarrow \mathbf{H}$ is limited to the path $\mathbf{G} \rightarrow \mathbf{B} \rightarrow \mathbf{G1} \rightarrow \mathbf{G2} \rightarrow \mathbf{C} \rightarrow \mathbf{H} \mid 4$. Notice that MH **F** is not able to reach MH **A** ($\mathbf{F} \Rightarrow \mathbf{A}$), since its shortest path to any GW is 3 hops which is above the k bound.

MHs take advantage of all nearby GWs to maintain connectivity. The number of available GWs is largely dependent on the placement of the GWs, MH density, and k. As we will see in Section 4, our routing protocols are designed to pro-actively ensure that a valid path to at least one GW is always known (if such a path exists).

[1] $\mathbf{S} \Rightarrow \mathbf{D}$ is read as "Source **S** connecting to destination **D**."
[2] $\mathbf{S} \rightarrow \mathbf{A} \mid 1$ stands for "**S** forwarding to **A** over 1 wireless hop."

CHAPTER 4

ROUTING IN K-HOP NETWORKS

The architecture of Chapter 3 calls for a routing protocol that is able to exploit the provided infrastructure when making routing decisions. In effect, the routing protocol should be "topology aware". In Figure 3.1, a link failure between MHs **H** and **C** would result in **H** losing access to the entire network. While this example represents the worst-case scenario, network connectivity largely relies on maintaining usable paths to nearby GWs. Therefore, the routing protocols we have designed share a common theme — keep GW paths as up-to-date as possible. It is quite possible for a MH to be within local range of several GWs at once. GWs take advantage of these multiple paths when forwarding packets. With mobility, "hand-offs" between different GWs can be made nearly seamless, as illustrated in Figure 4.1a through d.

In the figure, MH **A** is receiving packets from an outside source through GW **G1**, when it starts to move to the right (4.1a). As **A** moves farther away from **G1**, it establishes a 2-hop path to GW **G2** via MH **B**, while continuing to process packets from **G1** (4.1b). In 4.1c, **A**'s direct connection to **G1** breaks, and packets start to flow along the alternative path to **A** through **G2**. When **G1** detects the route failure to **A**, it performs route discovery and finds a new route to **A** through MH **C**. At this point, both paths are used to forward packets to **A**. Finally, in 4.1d, **A** comes in direct contact with **G2**, and forwarding along the path **G1** $\rightarrow$ **C** $\rightarrow$ **A** $\mid$ 2 ceases since a shorter path is available.

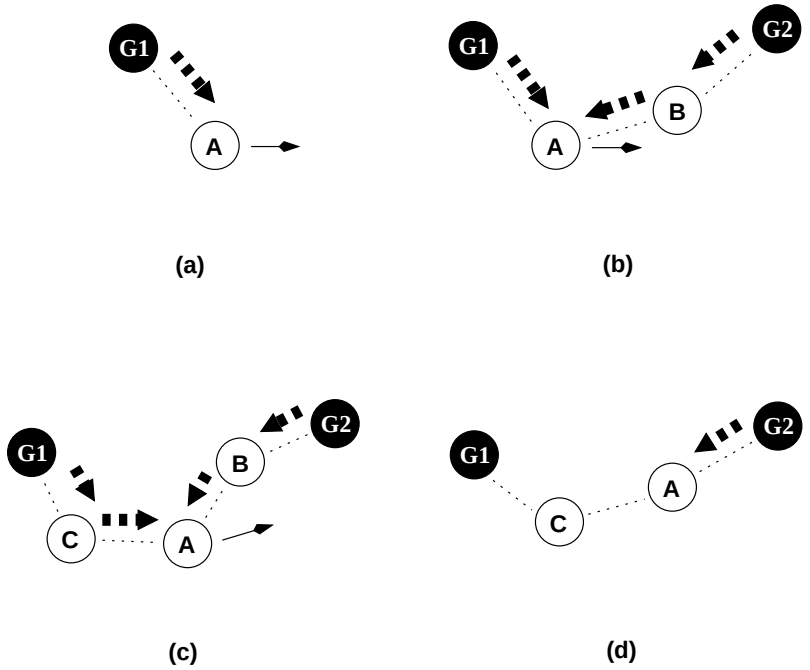

Figure 4.1 Smooth hand-off for MH **A** between GWs **G1** and **G2**.

Sections 4.1 through 4.4 describe the common elements of our two routing protocols KRP_1[1], and KRP_2. Details of KRP_1 are provided in Section 4.5, and in Section 4.6 for KRP_2.

4.1 Broadcast

A counter-based scheme is employed to reduce the amount of network broadcast traffic (see [61]). Although this scheme does not guarantee full coverage, we have seen it to be quite effective in our simulations at limiting the number of redundant retransmissions. The *drop threshold* of a MH refers to the number of identical broadcast packets a MH must overhear before suppressing its own transmission of the same packet. All simulations in this thesis are performed using a drop threshold of 4, based on the findings of Section 5.4. To further improve flooding in the network, any of the clustering algorithms mentioned

[1] k-hop Routing Protocol

in Chapter 1 could be utilized, although we did not explore this option. It is important to note that in some cases, our approach to mitigating the broadcast storm problem limits the ability to discover multiple paths to a destination (see Section 4.4). However, with a relatively dense network this situation is not as prevalent.

4.2 The Bulletin Board

Gateways communicate reachability of MHs in their local area via a shared *Bulletin Board* (BB). The BB consists of a set of destinations along with a list of GWs that are able to reach that destination. Also listed per GW is a hop count indicating the minimum number of hops that exist between the destination and the GW. As an example, the BB for the network in Figure 3.1 is provided in Table 4.1. When a GW discovers a new MH in its area, it adds its own address to the BB. Likewise, when a GW detects that a MH has moved out of its local area or is no longer reachable, it deletes itself from the BB for that destination. GWs are individually responsible for updating the BB immediately after learning new routing information. In essence, the BB serves as a global MH forwarding table for GWs.

Although in our simulations the GW resides in a shared memory space accessible by all GWs, in theory the BB can be realized in several forms. The first form consists of identical, separate copies of the BB maintained at each GW through simple message exchanges that specify changes to be made to the board. In this form, mechanisms are required to account for race conditions. The advantage of this form is that it is independent of the underlying network architecture, and hence it is the method of choice for our testbed (see Chapter 6). The second form of the BB requires an additional storage resource and a set of Remote Procedure Calls (RPCs) to handle access to the BB. A discussion of the two techniques as well as any optimizations are outside the scope of this thesis.

Table 4.1 Bulletin Board for the network of Figure 3.1

MH	GW	Hops
A	G1	1
	G2	2
B	G1	1
	G2	2
C	G2	1
	G1	2
D	G1	2
	G2	2
E	G2	1
	G1	2
G	G1	2
H	G2	2

4.3 The Gateway List

When a MH receives any packet containing a path pertaining to a GW, it takes this information and creates or updates the corresponding entry in its own *Gateway List* (GL). The GL serves as a repository for selecting a default path when no known route to a destination exists. Each entry in the GL contains just two fields — the gateway address and the minimum number of hops needed to reach the GW. The list is kept in sorted order so as to always prefer GWs that are closer to the MH. The GL is closely tied to a MH's routing table in such a way that all additional information about a GW listed in the GL (such as the next hop, the time remaining till the entry expires, etc.) are only stored in the routing table but can be quickly referenced from the GL by hashing on the address of the GW.

Entries in the GL can be removed in two ways. A route entry is deleted if the corresponding entry in the routing table expires. The amount of time allotted to a routing entry depends on the particular routing protocol being used; details on the timers are explained later in Sections 4.5.1 and 4.6.1. An entry can also be removed if the MH

receives enough notifications to invalidate all of the existing paths to the GW in the routing table.

A MH consults its GL when attempting to send a data packet to a destination that is either not listed in the routing table or is listed as invalid. In such cases, a routing table lookup is performed on the first entry in the GL to obtain the next hop information. To ensure proper forwarding by intermediate MHs, the data packet is encapsulated within a second IP header (IP-in-IP) that is addressed to the GW. The packet at this point is ready to be delivered. Intermediate MHs forwarding a data packet route solely on the destination address of the outermost IP header. These MHs drop packets that they cannot forward (no routing entry for the destination exists). The outer IP header is stripped off by the specified GW upon arrival, after which it is routed based on the true destination address. The purpose of the IP-in-IP encapsulation is to ensure that intermediate MHs do not inadvertently bypass the GW in the forwarding process based on their routing information for the destination.

4.4 Multiple Paths

Routes are acquired in KRP_1 and KRP_2 by processing the information stored in received control packets. These packets often contain a `<source, sequence number>` pair that identifies the original source of the packet, and the most recent *sequence number* for that source. Each MH and GW maintains its own monotonically increasing sequence number. Before sending a control packet, a source increments its sequence number and includes this number in the packet. When a downstream MH or GW receives the packet, the `<source, sequence number>` pair is compared against the stored sequence number for that source in the routing table. If the sequence number of the packet is greater than the one stored in the routing table, or if the sequence numbers are the same but the packet offers a shorter path to the source, the existing routing entry is updated to reflect the new next-hop for the source (set as the previous hop of the control packet) and sequence

number. This is identical to the sequence numbers of AODV [13, 14] In addition, the number of hops required to reach the source is recorded in the routing entry, and is supplied by the `hop count` field of the control packet. This field is incremented by one when sent or forwarded by a MH. GWs, on the other hand, either set (when sending) or reset (in the case of forwarding) this value to 1.

To extend our suite of routing protocols to support multiple paths per destination, we modify the existing route addition condition mentioned above. When the sequence number and hop count comparison is made, an advertised route (the one found in the control packet) is *added* to the existing path found in the routing table if the sequence numbers *and* hop counts are the same. In other words, multiple paths to a destination must possess identical sequence numbers, and contain the same number of hops to reach that destination. A path that is offered with the same sequence number as that of the routing table entry for the destination but a shorter metric (read: hop count) overrides any existing paths (in accordance with above).

When a MH or GW wishes to send data to a destination that is listed as having more than one path, the data packets are striped over the available paths. This technique enables the source to refresh each path independently upon successful transmission of a data packet. Since all paths are guaranteed to have the same metric, the inter-packet spacing at the receiving end remains fairly constant as compared to using paths with different metrics. A MH or GW that detects a link failure or receives a packet indicating a broken link will remove the offending path from the list of alternatives in the routing table, and switch to the next available path. The route is marked invalid once all of the paths have been removed.

The algorithm loses efficiency as k increases, since there is no method for determining path disjointness beyond the next hop the path is received on. For example, in Figure 4.2a, the two paths $\mathbf{S} \rightarrow \mathbf{B} \rightarrow \mathbf{A} \rightarrow \mathbf{G} \mid 3$ and $\mathbf{S} \rightarrow \mathbf{C} \rightarrow \mathbf{A} \rightarrow \mathbf{G} \mid 3$ to GW **G** will be accepted by MH **S** if k is 3, even though they share MH **A** as a common stop along the way. On the other hand, in Figure 4.2b, **S** will only receive one path to **G**, since **A**

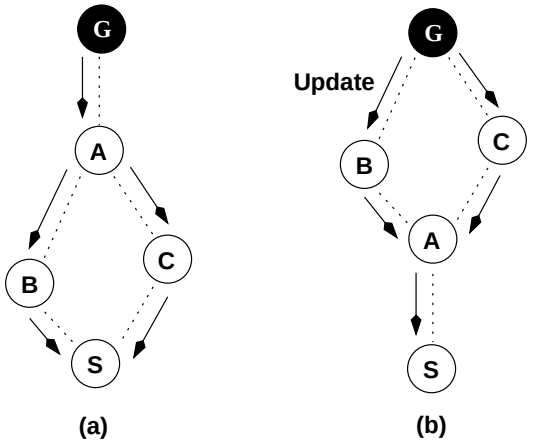

Figure 4.2 Multiple path creation issues.

will only relay the information once. Note, however, that **A**, upon experiencing a link breakage to either **B** or **C** in an attempt to transmit a data packet, will erase the broken path and utilize the secondary one, saving the packet from being dropped. This form of local repair is similar to that performed in DSR [11, 12].

4.5 KRP$_1$

The first routing protocol in the k-hop suite is KRP$_1$, based on the popular Ad Hoc On-demand Distance Vector (AODV) [13, 14] protocol. KRP$_1$ builds on top of AODV with the addition of default routes, a beaconing mechanism, and multiple paths. The major components of KRP$_1$ are detailed in the following sections. Unless otherwise stated, the behavior of KRP$_1$ follows that of AODV.

4.5.1 Beaconing

KRP$_1$ possesses a proactive *beaconing* mechanism used by GWs in order to track MH locations and hence reduce the scope of network-wide RREQs. The mechanism is aimed at reducing the number of RREQs performed by periodically recreating routes for ongoing

connections. To limit the scope of each beacon to only reach MHs that are within the local area, the Time-to-live (TTL) of each BEACON packet is set to k. BEACON packets also contain the beacon interval (`BEACON_INTERVAL`) of the GW, so that MH route expiration timers can be set appropriately (explained later in this section). Finally, the most recent sequence number of the GW is incremented and placed in the BEACON packet. BEACON packets are sent via broadcast. Upon receiving a BEACON packet, a MH will add/update a routing entry in its routing table to the originating GW according to the rules in Section 4.4. A MH will respond to BEACON packets if any of the following conditions are met:

Condition 1 *The GW is new to the MH*

Condition 2 *The route to the GW is expired*

Condition 3 *The MH is participating in a remote connection*

The first condition exists when a MH moves within range of a previously unknown GW. A reply is issued by the MH in this case to alert the new GW of its presence. In this case, only a single reply is sent, even though multiple paths to the GW may be discovered from the beaconing process. The second condition arises when a MH leaves and later returns to the local scope of the same GW. A MH sets each GW route to expire after a certain timeout period based on the number of allowed beacon losses (`BEACON_LOSS`) and the beacon interval of the GW (found in the BEACON packet). When a MH moves outside the local range of a GW, it is likely that the route will expire. Later, when the MH returns to the GW's area, it will receive a new beacon and validate the route, triggering a reply. As in the first case, only one reply is sent regardless of how many paths are formed to the GW. The last condition applies to MHs that are part of an active connection. In this situation, a MH replies along *multiple* paths if present (up to `MAX_MULT_PATHS`, a parameter) to the GW in order to establish a more robust connection. It is worthy of mention that for unidirectional connections, this step is not required by the source MH, since the reverse path is not needed. However this optimization for unidirectional

connections has not been implemented in any of our protocols. Finally, it is important to note that condition 3 takes precedence over that of conditions 1 and 2 when multiple conditions are satisfied. For example, if a MH hears from a new GW (condition 1) *while* participating in a connection (condition 3), it will reply multiple times rather than once, as it would for condition 1.

A received BEACON packet triggers a GL update according to Section 4.3. Non-duplicate beacons are scheduled for retransmission after a short random delay (maximum delay of 5 ms) or dropped if the TTL field of the packet is 0 (after being decremented). Before retransmission, the hop count field of the packet is incremented to account for the current hop.

Replies from a MH come in the form of an AODV-style RREP. The RREP specifies how long the GW should keep the route valid, and is unicast back to the GW using the most recently acquired path. The receiving GW handles the RREP by performing the following actions. First, the BB is updated accordingly (see Section 4.2). BB entries shared by GWs in KRP_1 have two additional fields that are modified during the update process. An *expired* field indicates that the GW was the last to hear from the MH, but its local route has since expired. All expired GWs are removed during the update, since their validity is questionable. The purpose of the *repair* field is explained in Section 4.5.3. A second action taken by the GW is to forward any data packets that might be waiting in the packet queue.

4.5.2 Route Discovery

Route discovery in KRP_1 is a multiphase process, starting with a source-initiated RREQ flood for the destination that is limited to k hops. If the destination is within this scope, it responds with a RREP along multiple paths as described in Section 4.4. A connection established in this fashion is considered *DIRECT*. Endpoints of direct connections *do not* reply to the periodic beacons of nearby GWs unless the GW indicated in the BEACON

packet is new or the routing table entry is expired (conditions 1 and 2 from Section 4.5.1). This behavior fulfills the tracking functionality of the beaconing mechanism while eliminating the need to keep the GW's path to the MH fresh (since the connection does not involve any GWs anyways). The exception to this rule is if the MH is servicing one or more remote connections in addition to the direct one.

If the initial RREQ attempt fails, the data packet is forwarded to the nearest GW as described in Section 4.3, and the MH switches to *REMOTE* state. The destination is added to a maintained list of unreachable destinations, in order to prevent the MH from performing a local search in the event of a future route failure. The destination can later be removed from this list if a direct connection is later established using overheard routing information, in which case the MH switches back to the DIRECT state. It is possible that a MH does not have a valid path to any GWs, in which case it issues a RREQ with the destination address set to the special `ALL_GWS` identifier. A GW receiving a RREQ with this address replies once with its own information via a RREP. The first RREP received by the MH triggers the release of all buffered data packets to the corresponding destination. The number of times a MH will attempt this step is based on the parameter `DISCOVERY_ATTEMPTS`.

A MH that loses connectivity with all nearby GWs and is forced to send a RREQ using the `ALL_GWS` address can potentially end up destroying one or more paths of nearby MHs when k is 3 or more. An example of this phenomenon is shown in Figure 4.3. In this 3-hop network, MH **S** has lost connectivity to GW **G**, and hence sends an `ALL_GWS` RREQ. This RREQ is received and forwarded by MHs **A**, **B**, **C**, and **D** to **G**. **G** responds to the two RREQs forwarded from **C** and **D** with RREPs. The arrows in the figure indicate the paths taken by the two RREPs. Since the RREPs from **G** contain a new sequence number, **A** is forced to erase any of its own existing entries for **G** (which includes the path through **C**) when processing the RREP for **S**. As a result, the multiple paths once available to **A** (acquired earlier) have been reduced to a single path, through **D**. This is much less likely to occur if intermediate MHs were able to reply to the RREQ on behalf

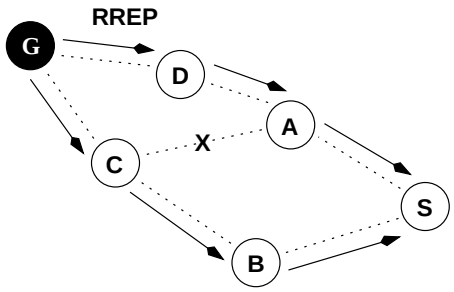

Figure 4.3 Valid paths can be destroyed as a result of an `ALL_GWS` RREQ.

of all GWs. However, allowing this type of behavior invites a series of replies when the network is dense and multiple GWs are nearby. Since paths are limited to a few number of hops and routes are updated regularly via beaconing, we decided it would be more beneficial to suppress intermediate replies (see [14]).

Route discovery is initiated by a GW when receiving a data packet that corresponds to a non-existent or expired BB route. If the GW is listed as the sole next-hop for the destination but the entry is marked as expired, the GW buffers the packet and performs route discovery for the destination. Destinations that have completely fallen off the BB trigger a network-wide *page* consisting of each GW issuing a RREQ for the destination with a TTL of k. Paging is an expensive mechanism, and hence the rest of the protocol is designed to reduce the amount of paging. The number of times the network will page for a destination is based on the `PAGE_ATTEMPTS` parameter.

4.5.3 Route Recovery

In KRP_1, route recovery is performed only by GWs and connection endpoints. This process is triggered when all of the paths to a particular MH or GW are rendered invalid. We first describe the steps taken by a GW to deal with such an event, followed by MH handling.

When a GW receives notification of a broken path that involves one or more routing entries, it attempts to re-route any pending data packets by handing them off to another GW listed in the BB for the destination. If no alternative GW is available, the GW sets the *repair* flag in the BB entry for the destination, and adds the destination to a list of MHs that require route discovery. Note that only one GW is ever repairing at any one point in time for a particular destination, since other GWs that experience a failure to the destination simply remove their listing when multiple GWs for the destination exist in the BB. This design decision resembles a "last man standing" approach to repairing connections, and was chosen in order to reduce the amount of overhead to the area around a single GW. Data packets are still forwarded to a GW even if it is repairing, and are buffered at the GW until the repair process is completed. If each destination has an alternative GW, no route discovery is necessary. When this is not the case, the resulting compiled list of destinations are copied into a RREQ packet, along with their most recent sequence numbers. The *multi-destination* RREQ is then broadcast locally by the GW. When the RREQ reaches one of the intended destinations, it responds normally with a RREP. The GW keeps track of those destinations that fail to respond, and follows up with an individual RREQ for the destination, up to `REPAIR_ATTEMPTS` times. After local recovery if a GW still cannot reach a destination, it activates the paging mechanism.

A RERR that eliminates the last path to a MH in a direct connection triggers a RREQ for the destination, following the rules of Section 4.5.2. If, on the other hand, the RERR pertains to a GW being used in a remote connection, the MH switches to the next available GW stored in its GL (see Section 4.3). If no GWs are available, a RREQ for `ALL_GWS` is issued. Data packets during the recovery process are buffered.

4.6 $\mathbf{KRP}_2$

The second routing protocol in the k-hop suite, KRP_2, is based on the Destination-Sequenced Distance Vector (DSDV) [18] protocol. The basic algorithm has been altered

to support multiple paths per destination (see Section 4.4) and employs a modified update scheme as described below. In addition, the packet format has been extended in order to distinguish between advertised routes that lead to GWs versus those that are for nearby MHs. Third, the selection criteria for advertised routes has been restricted so as to obey the k bound. Finally, the route dampening functionality (see [18]) has been omitted as a result of the k hop limit and the modified update behavior.

4.6.1 Updates

Both GWs and MHs participate in sending periodic updates offering reachability to nearby destinations. KRP_2 does not support partial updates (see [18]), except in the case of link failure. An update in KRP_2 *always* contains those MHs and GWs that are within $k-1$ or less hops from the source of the update. This is permissible due to the small scope of k. DSDV has been shown to perform much better when routes that sport a new sequence number are included in each update, rather than choosing to include only those routes that have changed metrics [6]. Routes in an update packet that correspond to a GW are denoted by a special *gateway* flag. A version of this flag also appears in the header of the packet, and is set whenever a GW issues an update. An update entry pertaining to a GW triggers a GL update as described in Section 4.3. When a MH or GW is selecting routes to include in the update, it may come across a routing table entry that has more than one path for the given destination. In this situation, the route is added only once, since the update entries do not carry any next hop information, only the associated metric, which is guaranteed to be the same across all of the paths when using the multi-path algorithm of Section 4.4. Updates perform a similar function to that of the beaconing mechanism in KRP_1, although in a more distributed and uniform fashion as MHs also participate in advertising their own presence. No replies are necessary to updates.

Due to the asynchronous nature of updates, it is possible that a MH or GW may send a data packet along a path that is longer than k hops. An example of such a scenario

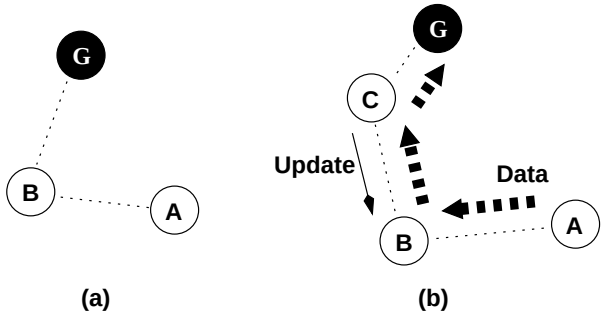

Figure 4.4 A temporary path outside the k bound.

is illustrated in Figure 4.4, where k is 2. In Figure 4.4a, MH **A** has a stored route to GW **G** through MH **B**. The recorded length of this route is 2. Likewise, **B** knows it can reach **G** in one hop. A short while later, in Figure 4.4b, **B** does not receive an update from **G** as it expects to, but instead receives an update from another neighboring MH **C**, advertising a route to **G** with 2 hops along with a new sequence number for **G**. **B** updates its routing table to accept the new path. At this point in time, if **A** were to send a data packet to **G**, the path taken would be $\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{C} \rightarrow \mathbf{G} \mid 3$, which is longer than k. This situation is resolved when **A**'s route to **G** times out, since **B** will no longer include **G** in its own updates. Such events are allowable, since their existence is transient.

4.6.2 Link Failures

Detected link breakages in KRP_2 are handled by the immediate broadcast of an update containing the broken route. This update also includes any other routes that have changed since the last update. Each MH and GW that hears this triggered update will invalidate their own entry to the newly unreachable destination *if and only if* the source of the update happens to be the next-hop recorded in the routing table for the destination. This is in contrast to normal DSDV operation, in which the destination is invalidated regardless of the source. With this modification, unnecessary invalidations

are avoided. The drawback to this approach is that the destination is unlikely to overhear the update packet signifying the link breakage, and therefore will not issue an immediate update to replace the broken paths. However, with multiple paths for each destination, we anticipate this limitation to be mitigated.

4.6.3 Forwarding

Data packet forwarding in KRP_2 follows standard next-hop forwarding rules. Local paths are preferred over ones that pass through a GW in order to alleviate the bottleneck at the GW, even if a shorter path exists through a GW (for $k \leq 3$). For larger k, the path through the GW may be chosen (out-of-scope for this thesis). When a local path does not exist in the routing table, the MH utilizes a default path as indicated in Section 4.3. A MH that does not have any available local or default paths to the destination indicated in the data packet will buffer the packet for `UPDATE_INTERVAL` seconds, the length of time between successive updates.

CHAPTER 5

SIMULATION

This section details the experiments that were performed to test the routing protocols of Chapter 4.

5.1 Simulation Environment

Simulations were completed using the Network Simulator (NS) 2 [62] with wireless extensions from the Monarch Project [63]. All runs were performed with version 2.1b9a of the simulator. The MAC[1] protocol used by NS 2 is modeled after the IEEE 802.11 DCF[2] mode [64]. The radio propagation model employed is two-ray ground. The link bit-rate is set at 2 Mb/s. The maximum transmission range of MHs and GWs is 250 m. Mobility in all scenarios is modeled using the "random way-point" model [6]. Pause times are not included in any of the simulations and so are set to 0. Each simulation runs for a duration of 900 seconds. A short summary of parameters is presented in Table 5.1.

For all experiments, we chose 100 MHs to roam around a rectangular area of 2400 m $\times$ 600 m, using for the most part four different maximum speeds of 1, 5, 10 and 20 m/s. GWs are placed 600 m apart and 300 m from the top and bottom of the rectangle, as indicated in Figure 5.1. To cover a wide range of mobility and traffic patterns, for each

[1]Medium Access Control
[2]Distributed Coordination Function

Table 5.1 Simulation Parameters.

Transmission Range	250 m
Simulation Duration	900 s
Mobility Model	Random way-point
Pause Time	0 s
Medium Access Protocol	IEEE 802.11 DCF
Link Bit-rate	2 Mb/s

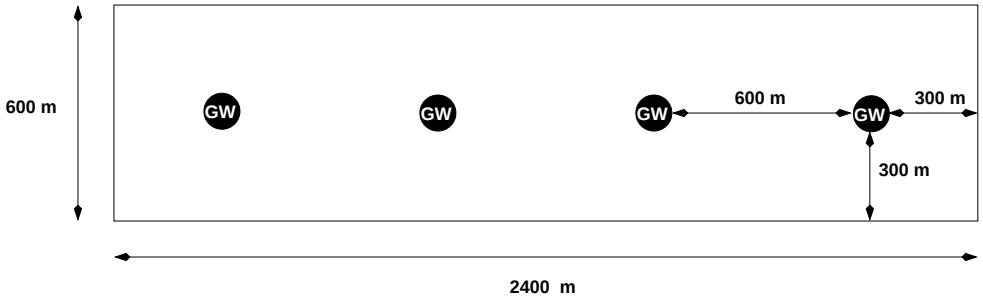

Figure 5.1 Placement of GWs in the simulated area.

point of data we ran 36 simulations consisting of 6 different mobility patterns and 6 traffic files based on the communication model of Section 5.3. A 95% confidence interval was computed for each data point, and is indicated in each graph by vertical bars. These lines are labeled as "error" graphs in the legend.

5.2 Metrics

The overall goal of our experiments is to measure and compare the ability of the proposed routing protocols in the k-hop suite to react to a changing network topology and provide seamless connections between communicating MHs. To accomplish this goal, we subject them to a series of different scenarios representing a range of conditions. The performance of our routing protocols are compared based on the following metrics.

1. **Packet Delivery Ratio** — The ratio between the number of packets originated by a source and the number of packets received by the destination.

2. **Average End-to-End Delay** — The time taken for a data packet to reach its final destination after being sent by the source.

3. **Routing Overhead** — Measured as number of transmitted kilobits of control packets per second per node (MH and GW) in the network.

4. **Average Path Length** — The number of wireless hops a data packet traverses from source to destination.

5.3 Communication Model

Communications between MHs in our simulations are modeled using Constant Bit Rate (CBR) traffic sources and sinks. By increasing the number of CBR sources, we are able to observe each protocol's ability to make more frequent routing decisions. The intention of our communication model is not to test the routing protocol's operation under a significant network load, therefore, each CBR source is instructed to send 64 byte packets with 200 ms spacing or an equivalent rate of roughly 2.5 kb/s. This traffic load is similar to that used in previous tests of popular routing protocols [6]. Connections are started at times uniformly distributed between 30 and 60 seconds. The first 30 seconds of each simulation are set aside to allow the network to settle and to populate routing tables. Finally, each CBR source and sink in our simulations are distinct. This provides a more challenging environment for the routing protocols, as there is no overlap between sources and sinks.

5.4 Protocol Parameters

As indicated in Chapter 4, KRP_1 and KRP_2 have several parameters that are variable. As such, we wish to test the protocol performance under different parameter values; therefore we have designed and carried out the experiments of Section 5.5. One parameter that is fixed throughout all experiments is the drop threshold (`DROP_THRESHOLD`) from Section 4.1. To select a suitable value for this parameter, we conducted a short experiment involving KRP_1's beaconing mechanism, which begins with a GW broadcast. In the experiment, `DROP_THRESHOLD` was varied from 2 to 5, and the broadcast performance was then compared to broadcasting without a drop threshold. Specifically, the number of replies by MHs to GW BEACONs were recorded, as well as the number of those replies that were actually received by each GW. Each test was run for the whole 900 seconds. Figure 5.2 shows the results of this experiment. The solid line of the figure indicates the number of replies that were sent to the GWs as compared to those sent with no threshold for broadcasting. As can be seen, a drop threshold of 4 allows 99% of the packets to be delivered successfully. The dashed line in the figure illustrates the percentage of those RREPs for each drop threshold that were received by GWs. Since a drop threshold of 4 is over 99% in both measurements, it was therefore selected for the parameter `DROP_THRESHOLD`.

5.5 Results

Several experiments were devised in order to test the performance of the routing protocols when certain parameters are modified. Before looking at the results, it is worth noting that the data presented, particularly in terms of packet delivery ratio, is an understatement of the actual protocol performance. The reason for this is as follows. Recall from Chapter 3 that a MH is part of the k-hop network *if and only if* it can reach a nearby GW using k wireless hops or fewer. Since MH locations are randomly generated, it is often the case where either the source or destination of a connection is outside the

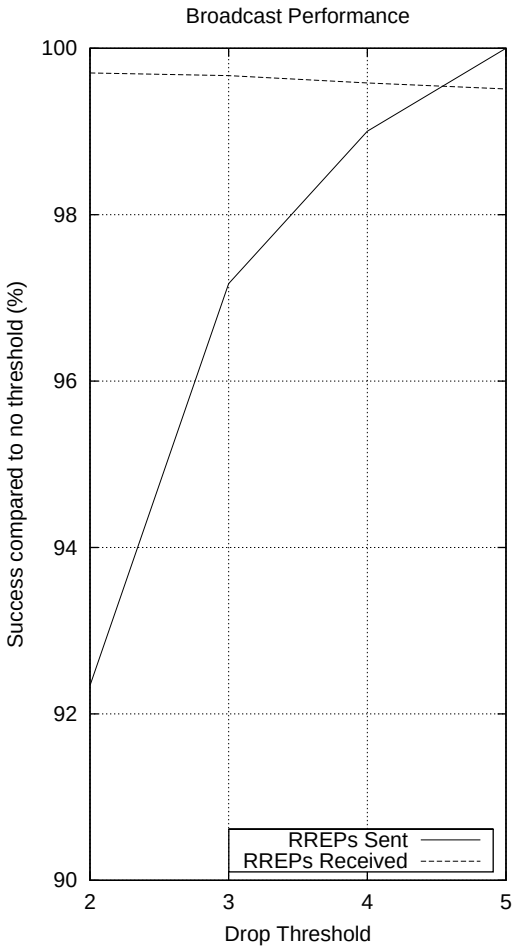

Figure 5.2 Broadcast performance under different drop thresholds.

bounds of the network. In such a situation, a significant number of data packets will be dropped by either the source or the GW. In KRP_1, if the source is outside the k bound, it will repeatedly search for an available GW as described in Sections 4.5.2 and 4.5.3 before admitting defeat. In this case, the MH will refrain from searching until it hears a BEACON from a GW. Likewise, a GW that cannot find a viable path for a destination will give up after a series of unsuccessful pages. After this point, the GW will check for the MH's presence upon receiving a packet destined for it, but will not take further action until the MH replies to a BEACON and gets entered into the BB. For KRP_2, in both situations packets are temporarily held if no route exists. If the associated timer

for the packet expires and still no route is available, the packet is simply dropped. It may be argued that these dropped packets should not necessarily be counted toward the routing protocol's ability to successfully deliver packets. However, identifying such packets before, during, or even after the simulation is a difficult task, and time did not permit for the filtering of these occurrences. As such, the results are instead comparative and serve as an indicator of the degree of connectivity of the network; a high success rate symbolizes high connectivity, and vice versa.

5.5.1 Experiment 1: The Effect of Multiple Paths

In this experiment, we wish to study the impact of our multiple path algorithm on routing performance for both KRP_1 and KRP_2 by varying the parameter `MAX_MULT_PATHS`. In particular, for KRP_1, we expect to see a decrease in the amount of overhead generated, due to the reduced number of RREQs necessary when an alternative route is available. For KRP_2, we anticipate a higher delivery ratio for multi-path over single-path, as more path alternatives are available between updates. k in all of the simulations is fixed at 2. `BEACON_INTERVAL` for KRP_1 is set to a wide interval (10 seconds in this case) in order to increase the frequency of route changes experienced by the network. The same interval was chosen for updates in KRP_2 (`UPDATE_INTERVAL`).

Figures 5.3 through 5.6 show the results of varying the number of allowable paths from 1 to 3 for a single destination for both protocols. These results are plotted against the aforementioned mobility rates.

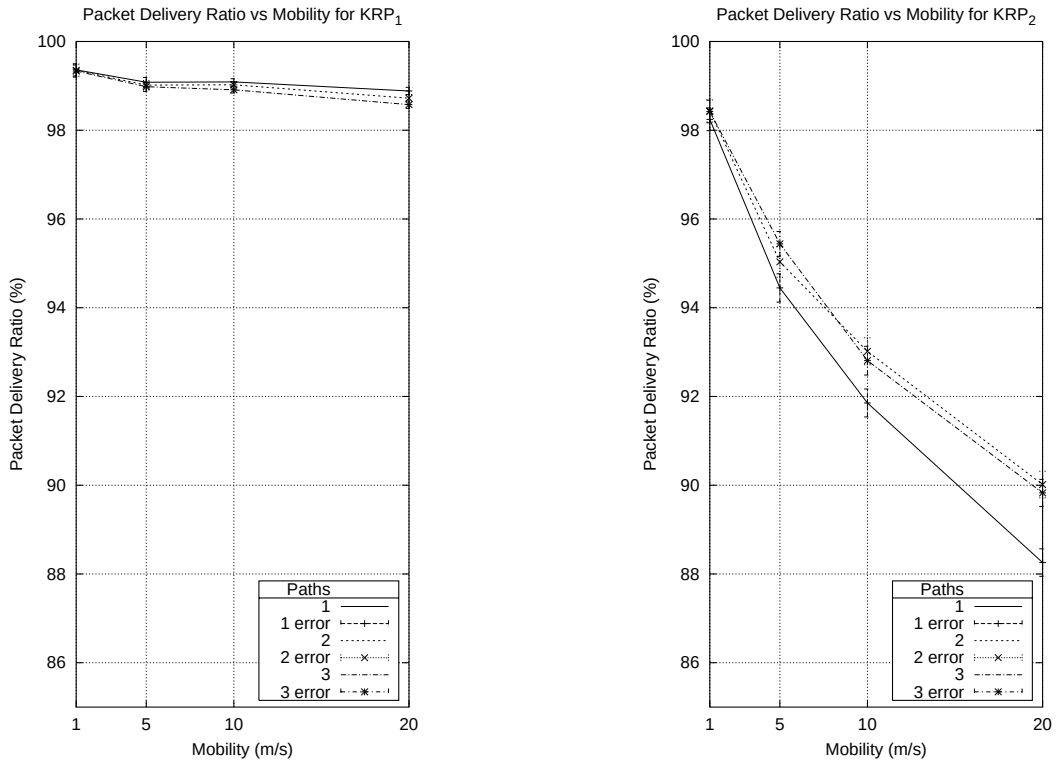

Figure 5.3 Packet Delivery Ratio for KRP_1 and KRP_2 for Experiment 1.

Figure 5.3 indicates that the success rate for KRP_1 is not largely affected by the multiple path feature, in fact, it is better off without it. This is partly attributed to the reactive nature of the protocol; broken paths are immediately fixed by the RREQ/RREP process when needed. Additional analysis the behavior seen is discussed in Section 5.6. On the other hand, a slight improvement is seen in KRP_2 when the number of multiple paths is increased from 1 to 2. Some insight as to why a better benefit is not achieved is discussed in Chapter 7.

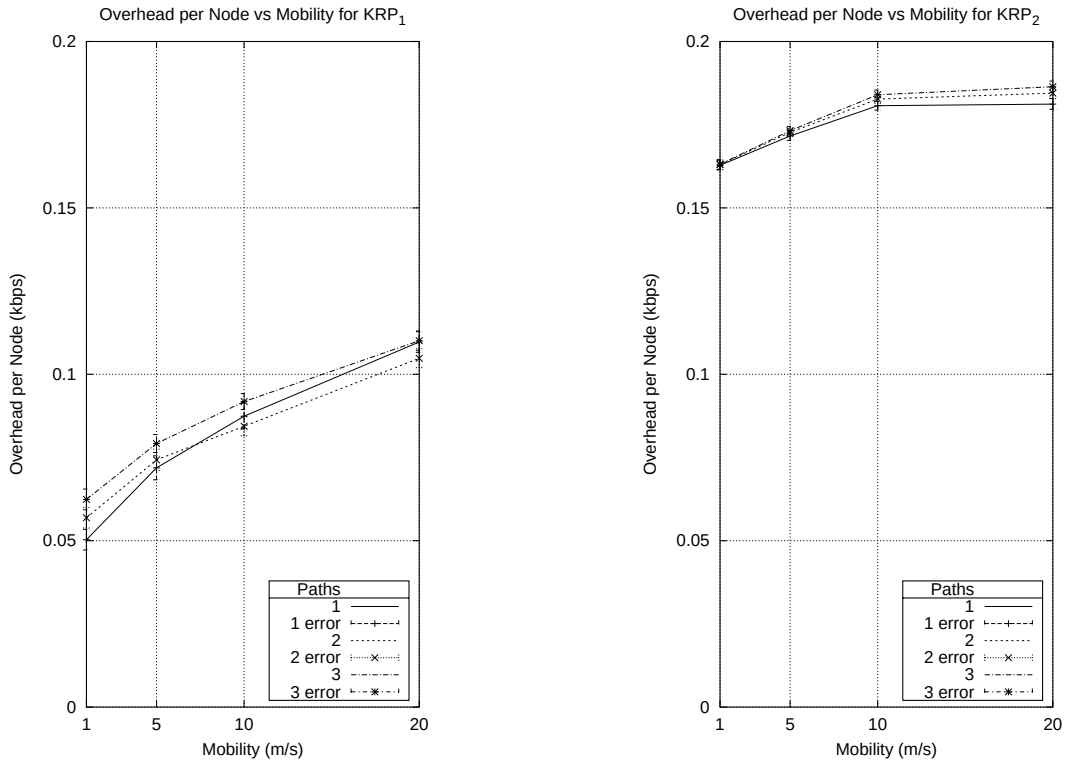

Figure 5.4 Control Overhead for KRP_1 and KRP_2 for Experiment 1.

In Figure 5.4, the curves for overhead of KRP_1 do not distinctively decrease as expected for increasing multiple paths. Further analysis reveals that the cost of maintaining the multiple paths in the form of multiple RREPs counteracts the benefit received from having more than one path choice. In particular for the value of k used, paths are short enough that the cost to execute route recovery between a MH and GW is nearly identical to the additional overhead needed to support multiple paths. For KRP_2 as well, there is no significant overhead difference between the different path versions. However, this is to be expected, as the multi-path feature of KRP_2 does not require any additional packets for support.

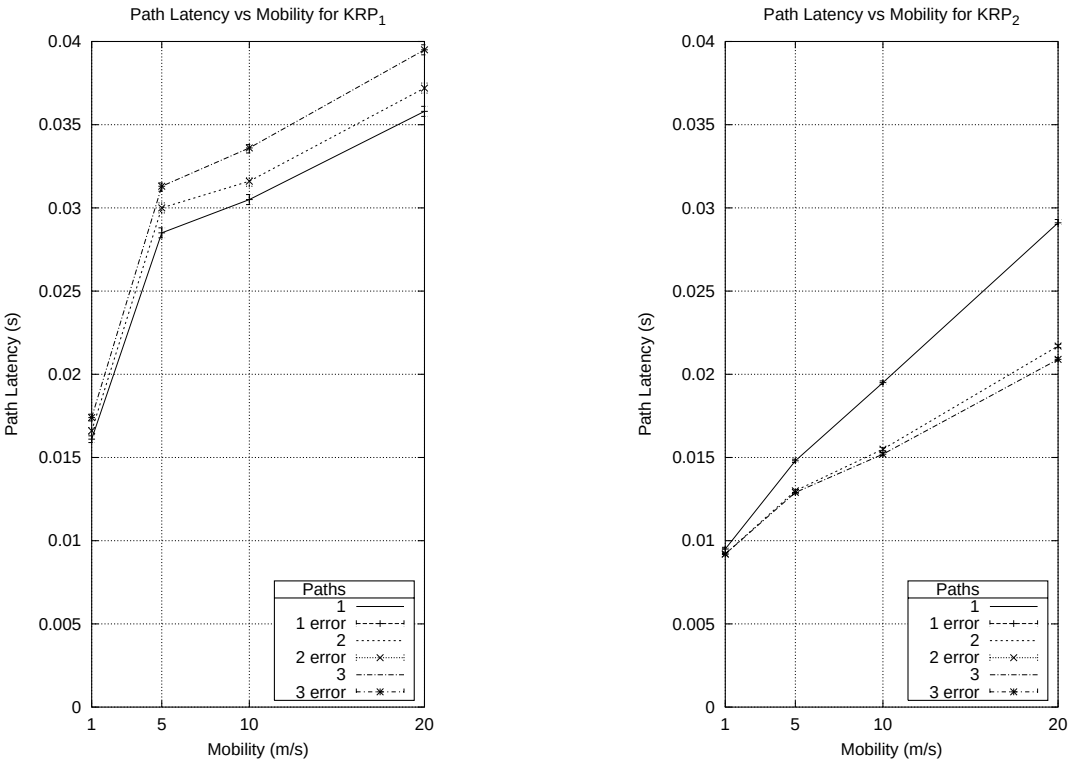

Figure 5.5 End-to-end Packet Latency for KRP_1 and KRP_2 for Experiment 1.

Figure 5.5 depicts that the average end-to-end path latency increases with the introduction of more than one allowable path. Extra time is spent trying alternate routes, with limited success. The link failure detection time is a key contributor to additional latency. This effect is not as pronounced in KRP_2, given that failure of multiple paths is not followed up by a new route discovery effort.

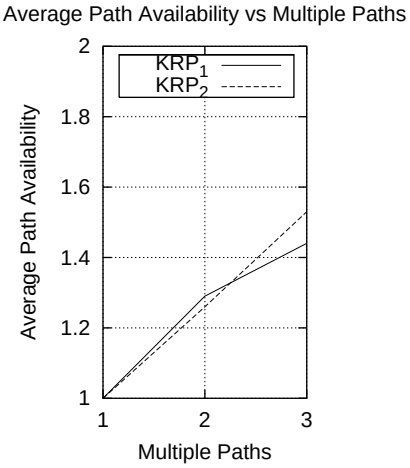

Figure 5.6 Average number of available paths during Experiment 1.

Finally, the average number of paths per destination when sending or forwarding packets is shown in Figure 5.6 for KRP_1 and KRP_2. These averages are affected by several factors, including the drop threshold used (see Section 4.1) MH density, and GW placement, among others. It is important to note that the graph includes average path availability for MHs that are within direct contact of a GW, which is always 1. Therefore, the values of these samples bring down the average for those MHs that are more than one hop away from a GW.

This experiment indicates that the optimal value for `MAX_MULT_PATHS` for KRP_1 is 1, whereas for KRP_2 this parameter should be 2. These values account for the best packet delivery ratio at a reasonable overhead cost, while offering a suitable average path latency. As a result, the remaining experiments are performed using these values.

5.5.2 Experiment 2: The Effect of Beaconing and Update Intervals

Our second set of simulations illustrate the effect of varying beacon and update intervals in KRP_1 and KRP_2 respectively. Decreasing the intervals should increase the amount of

protocol overhead while potentially increasing the packet delivery ratio and decreasing the average end-to-end delay.

The results in Figures 5.7 through 5.9 demonstrate the impact of beacon and update intervals of 3, 5 and 10 seconds for KRP_1 and KRP_2 against the various mobility rates.

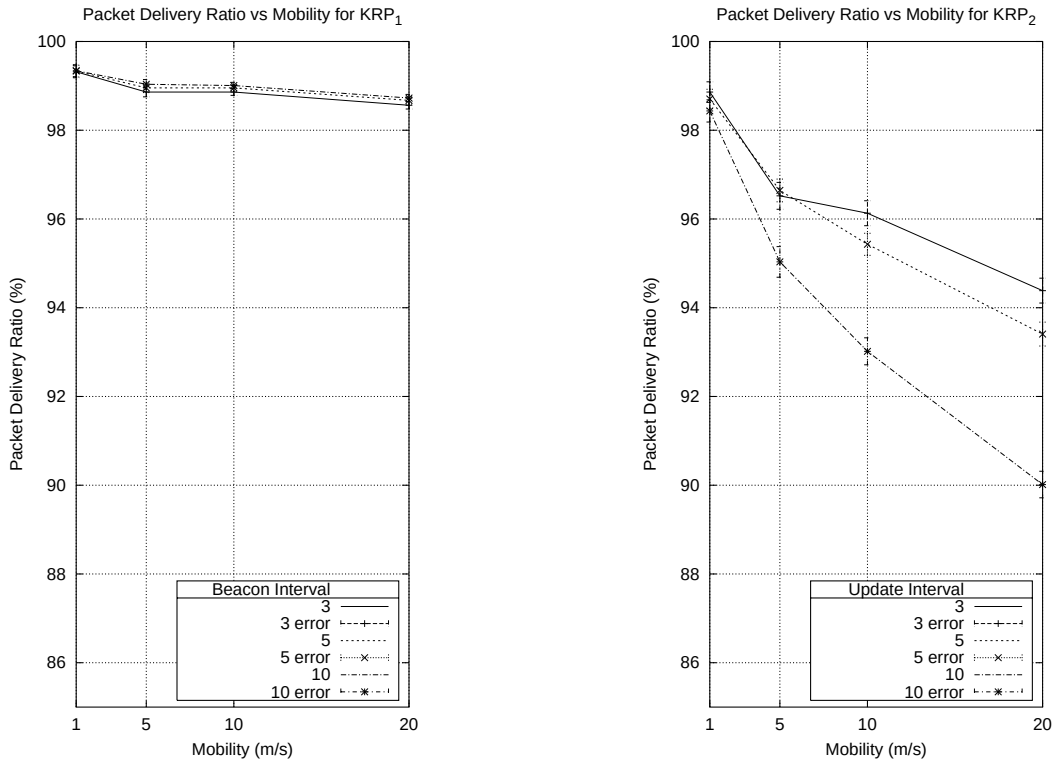

Figure 5.7 Packet Delivery Ratio for KRP_1 and KRP_2 for Experiment 2.

An interesting event is revealed in Figure 5.7 for KRP_1. According to the graph, the protocol achieves a slightly higher (fractions of a percent) packet delivery ratio when the beacon interval is more spread apart (10 seconds) rather than close together (3 seconds). An in-depth examination reveals that in fact the higher beacon intervals result in fewer pages, RREQs, and a higher percentage of available paths. This leads us to believe that KRP_1 performs better when left only with reactive mechanisms for repairing broken links, rather than actively setting up new ones. See Section 5.6 for a more detailed explanation in support of this trend.

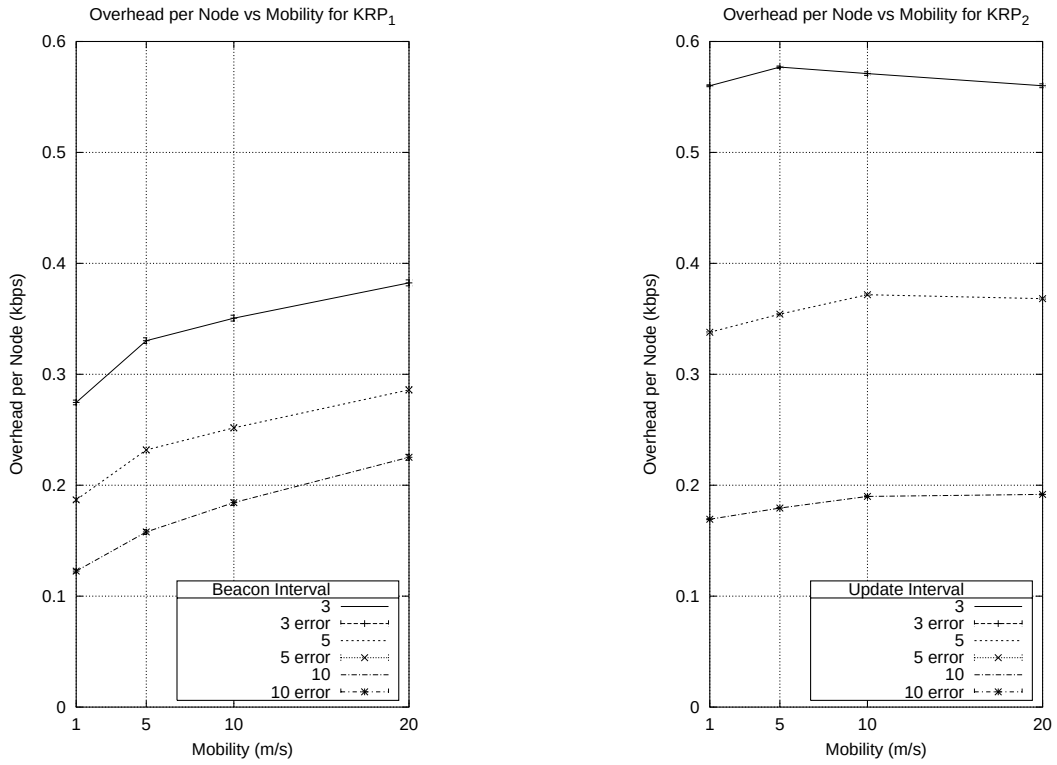

Figure 5.8 Control Overhead for KRP_1 and KRP_2 for Experiment 2.

The curves of Figure 5.8 denote the increase in overhead as the beacon interval or update interval becomes smaller. Note that the rise in overhead is not linear. The difference in overhead between intervals 3 and 5 is more significant than that of between intervals 5 and 10, even though the first interval is less. This is easily understood however when considering that the overhead is roughly halved when shifting from an interval of 3 seconds to 5 seconds, and then halved again when using a value of 10 seconds.

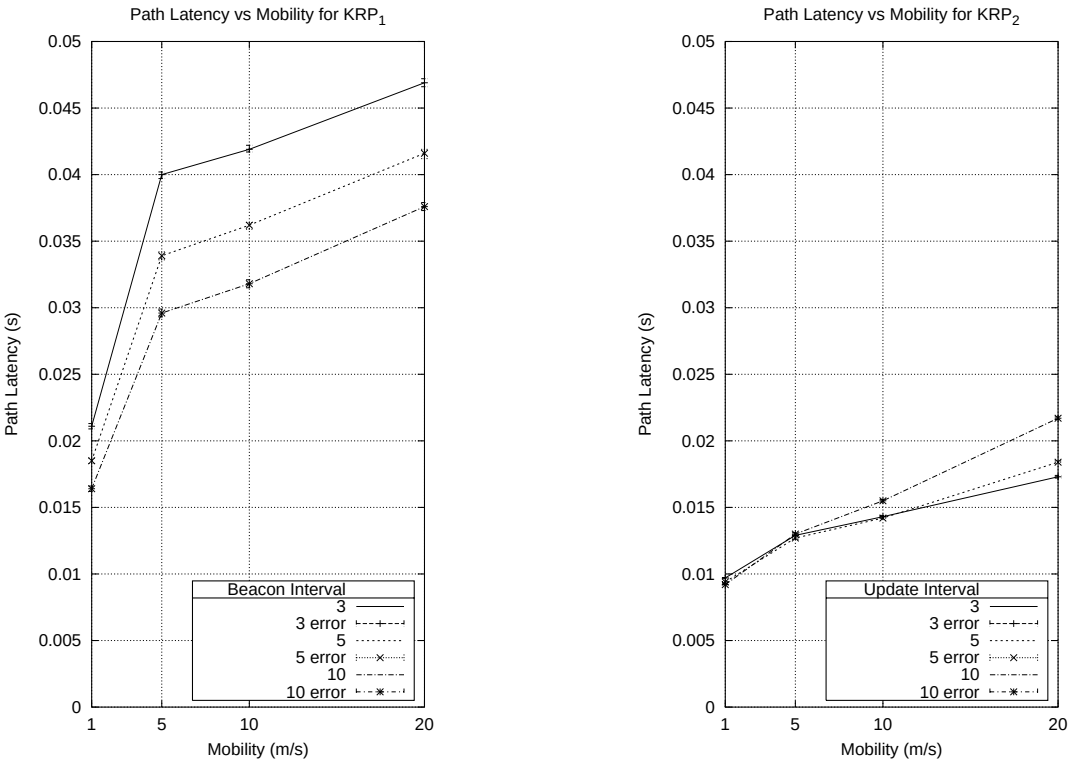

Figure 5.9 End-to-end Packet Latency for KRP_1 and KRP_2 for Experiment 2.

KRP_2 holds the advantage in terms of end-to-end packet latency as evident in Figure 5.9. Of importance in this figure is that it shows packet latencies *increasing* as the beacon or update interval *decreases* for KRP_1. Indeed, the additional overhead causes more packet retransmissions resulting in longer path delays, among various other reasons mentioned in Section 5.6.

5.5.3 Experiment 3: k and the Connectivity/Overhead Trade-off

In our last set of simulations we vary k to observe its impact on network connectivity and overhead. All simulation parameters are the same as in the previous two experiments, except that k is now varied from 1 to 3 rather than fixed at 2.

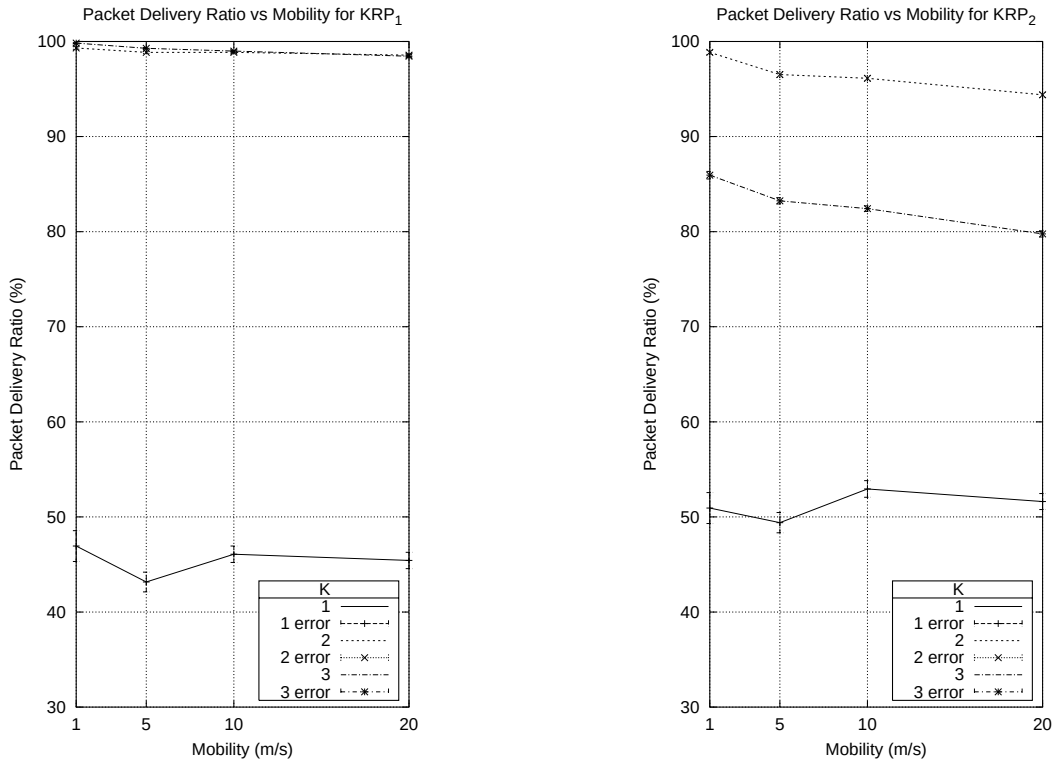

Figure 5.10 Packet Delivery Ratio for KRP_1 and KRP_2 for Experiment 3.

The packet delivery ratio of KRP_1 and KRP_2 is shown in Figure 5.10. Clearly, KRP_1 outperforms KRP_2 along all mobility speeds. Recall that the low ratio for both protocols when k is equal to 1 is primarily due to MHs leaving and entering the network (within 1 hop of a GW). When k is increased to 2, the percentage of MHs that are consistently connected dramatically rises with k=3 achieving near continuous connectivity. Table 5.2 lists the average connectivity experienced by MHs for each value of k for each protocol. These averages were obtained by observing the number of entries in a MH's GL during every simulation second beyond the startup period. If more than one GW is present in the list, the MH is connected. For KRP_2 in the figure, the success at which the protocol delivers packets best is $k = 2$. As paths get longer, more routes break which are not repaired until the source or destination MH issues its next update. With the current implementation, KRP_2 is limited to adequately serving 2-hop networks.

Table 5.2 Average MH connectivity for $k = 1, 2$ and 3 in Experiment 3.

K	$\mathbf{KRP}_1$	$\mathbf{KRP}_2$
1	66.62	68.82
2	98.80	98.87
3	99.81	99.73

The numbers from Table 5.2 in conjunction with the overhead graphs of Figure 5.11 offer support for our early claim regarding the connectivity and overhead behavior over as k increases. Diminishing returns are observed for connectivity, while the overhead follows super-linear growth. However, this result is limited by the understanding that both KRP_1 and KRP_2 possess pro-active features that do not adjust based on k. The increase in overhead for a purely reactive protocol may fair better in this metric.

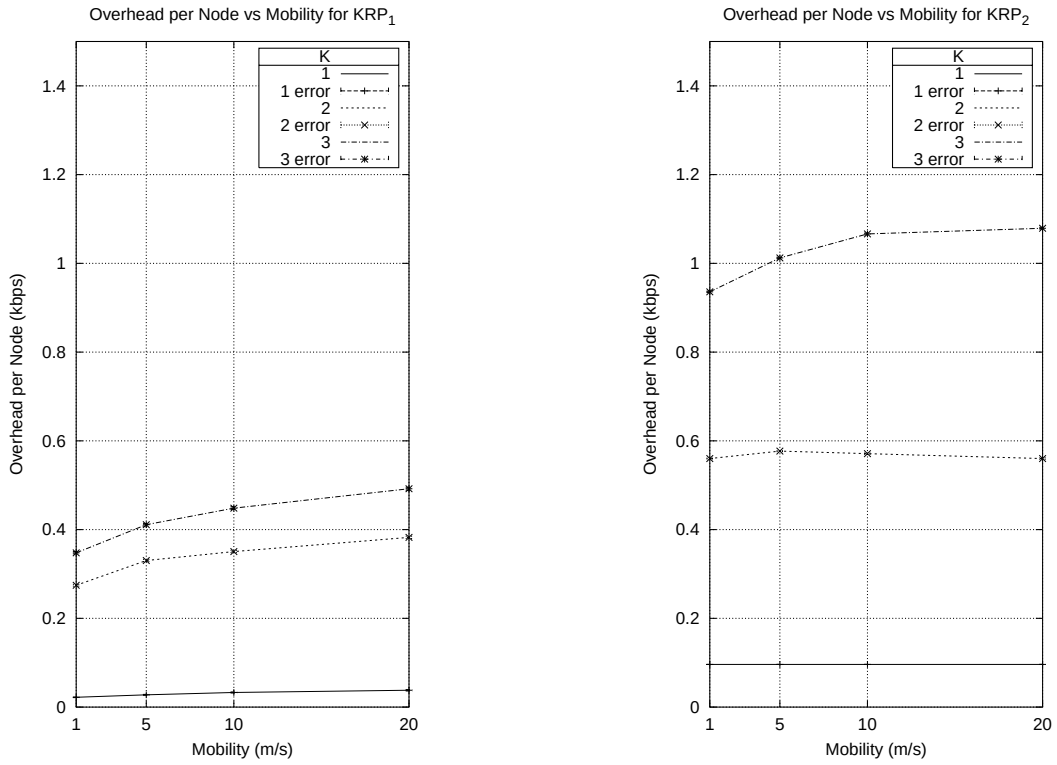

Figure 5.11 Control Overhead for KRP_1 and KRP_2 for Experiment 3.

Figure 5.11 depicts the routing overhead experienced over the range of mobility rates. For both KRP_1 and KRP_2, the overhead grows by nearly a factor of 10 between $k = 1$ and $k = 3$. This is attributed to the high cost of forwarding BEACONS in KRP_1 and the quick growth of update packets in the case of KRP_2. As further proof, for $k = 1$ the average size of an update packet for KRP_2 registers at 8 bytes (just the header with no entries), while the update packets transmitted when $k = 3$ carry a weight of 343 bytes (roughly 41 entries).

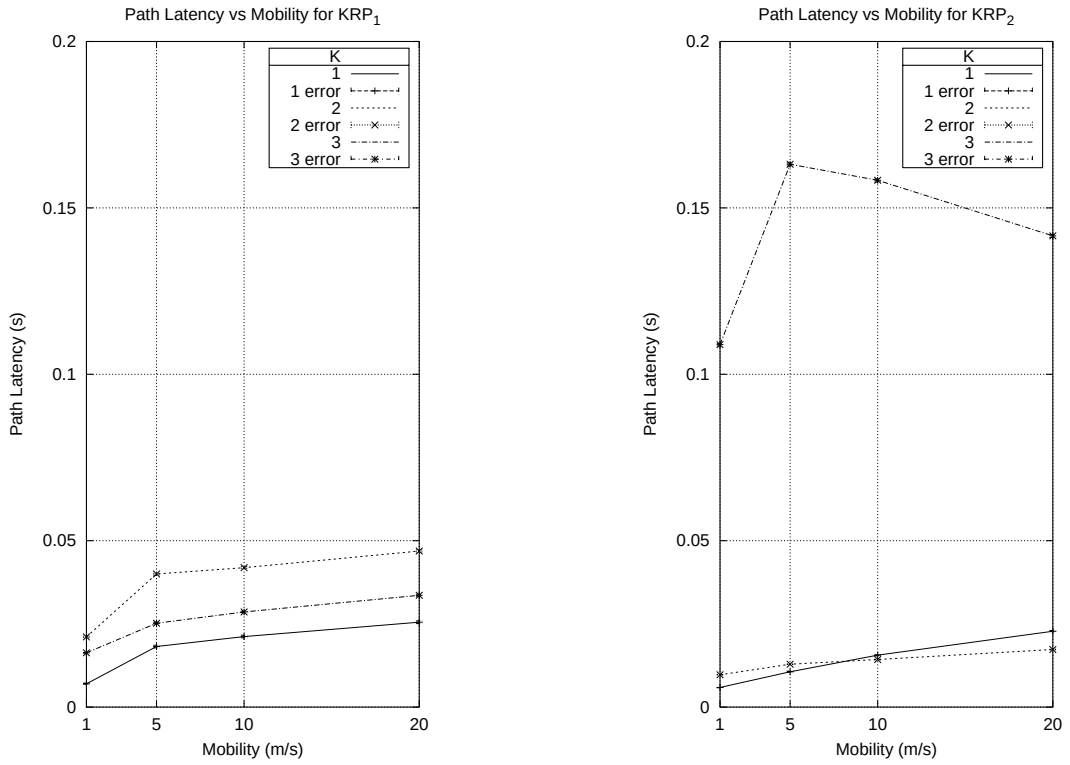

Figure 5.12 End-to-end Packet Latency for KRP_1 and KRP_2 for Experiment 3.

The interesting aspect of Figure 5.12 is the latency of packets for KRP_2 when k is 3. As can be seen, the average packet latency is an order of magnitude larger than that of $k = 2$. Upon inspection, this phenomenon is a result of the protocol's inability to converge under the given mobility rates. Without convergence, packets run the risk of being shuffled back and forth between MHs before eventually reaching their destinations or being dropped.

With KRP_1's pro-active beaconing feature, it may be expected that the packet latencies for the protocol closer match that of KRP_2. This is not the case due to several reasons. First, recall that from Section 4.5.2, KRP_1 begins with local route discovery for a destination before forwarding packets to the nearest GW. This process incurs extra latency for the initial packets of the connection. Second, a GW that receives a data packet for a destination that has not been used recently but is listed in on the BB requires a response

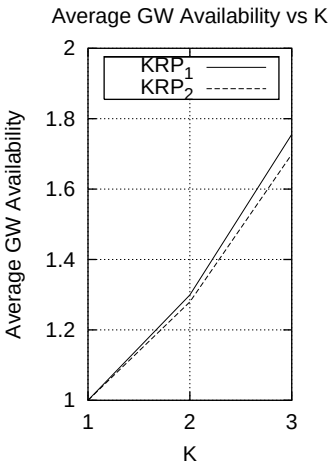

Figure 5.13 Average number of available GWs during Experiment 3.

Table 5.3 Maximum GW availability during Experiment 3.

K	$\mathbf{KRP}_1$	$\mathbf{KRP}_2$
1	1	1
2	2.00	2.39
3	3.00	2.45

from the destination to a GW-initiated RREQ before continuing. Finally, link breakages still trigger a new route discovery cycle for the destination, during which packets are temporarily buffered.

Figure 5.13 shows the average number of GWs that are present in a MH's GL during the simulations. The graph also reflects the average number of GWs listed on the BB for a destination when making forwarding decisions. As k increases, GW availability increases, as the local scope of both MHs and GWs increase. A higher average affords the network more opportunities for successfully forwarding packets. The maximum GW availability observed for each value of k is listed in Table 5.3.

5.6 Analysis & Optimizations

The results of the previous section for KRP_1 at times appear counter-intuitive, and hence require some additional exploration to provide a more detailed explanation of its behavior. As such, this section offers some insight on KRP_1 based on further experiments. We also describe some optimizations that can help improve the performance of the protocol.

Upon inspection of the simulation trace files, a few interesting situations appear that adversely affect KRP_1's performance. Most notably, increasing the beaconing frequency does not necessarily improve the quality of selected paths in the network. In fact, there is a high occurrence of instances where a MH selects a different next-hop via a new BEACON only to have the path break shortly afterward. Often in this case if the MH has no alternative path, it will issue a RREQ for `ALL_GWS`, and return to using its previous next-hop (i.e., the one it was using before it received the BEACON). This seems to suggest that frequent "updates" can destroy a number of already stable connections. Hence, we are led to believe that it is more appropriate to deal specifically with problematic paths rather than refreshing all paths at once. The additional path failures accounts for an increase in latency and a decrease in the success rate as the beacon interval becomes shorter.

The added latency is also attributed to the act of beaconing itself. During the beacon and reply process, data packets collide with the broadcast BEACON packets, requiring retransmission. They also must contend with the numerous replies that are sent, especially when the number of allowable multiple paths increases. Our communication model specifies that up to 60% of the network will respond one or more times to every BEACON received, assuming full connectivity. This short burst of periodic control overhead puts a strain on the network that is reflected in the data packet latencies, such the ones plotted in Figure 5.14. This graph shows the time interval of 120 seconds to 126.5 seconds for a run with `BEACON_INTERVAL` set to 3. Notice that around the time that beacons and replies are sent (at 120, 121.5, 123, 124.5 and 126 seconds) the end-to-end packet latency skyrockets to nearly 10 times that of the average latency in-between the beacon periods.

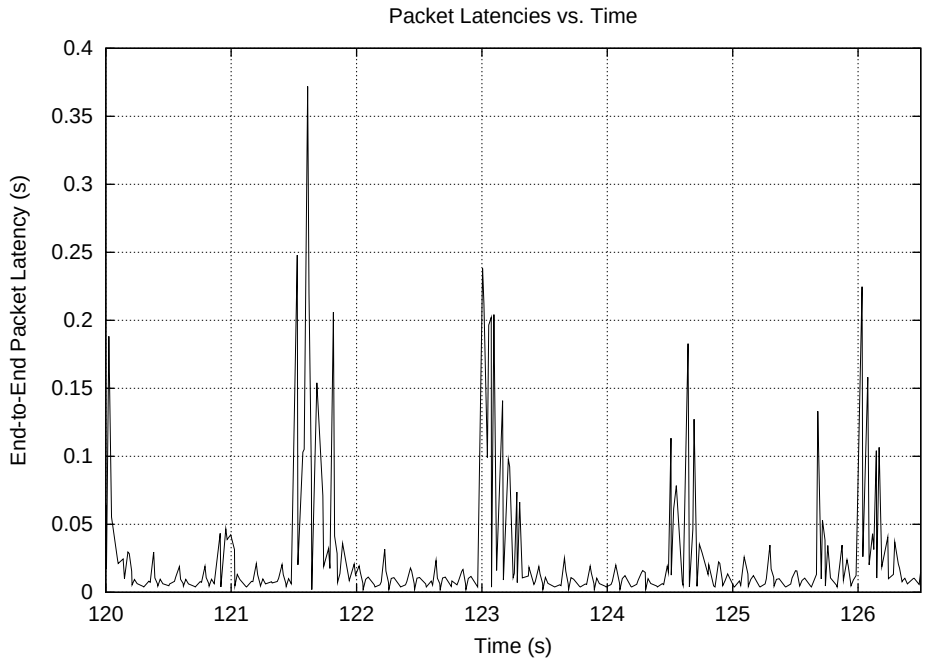

Figure 5.14 End-to-End Packet Latency vs. time for KRP_1 (`BEACON_INTERVAL = 3`).

Consequently, the less often beaconing is performed, the lower the overall packet latencies will be.

To support this conclusion, Experiment 2 was performed again with only 10 connections. The mobility has also been extended to include 40 m/s. At this high mobility rate, we expect to see a more pronounced difference in the performance given the various beacon intervals. All other parameters are identical. Figures 5.15 and 5.16 show the packet delivery ratio and latency of KRP_1 under less traffic.

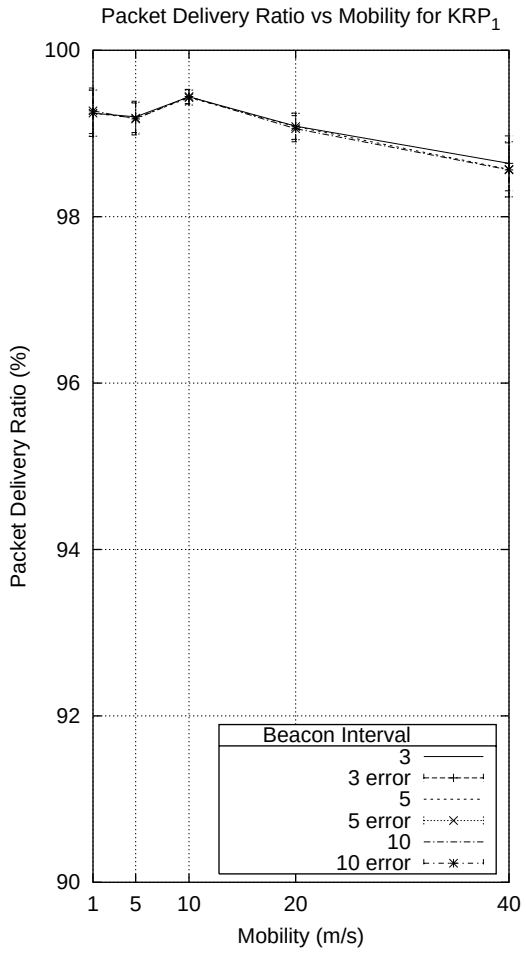

Figure 5.15 Packet Delivery Ratio for KRP_1 with 10 connections.

As can be seen from Figure 5.15, the packet delivery ratio with fewer traffic sources increases (rather than decreases as was the case earlier) as the beacon frequency is increased, although the gains are marginal. Likewise, the average end-to-end latency in Figure 5.16 for a beacon interval of 3 seconds is now less than that of the 5 and 10 second intervals at higher mobility speeds.

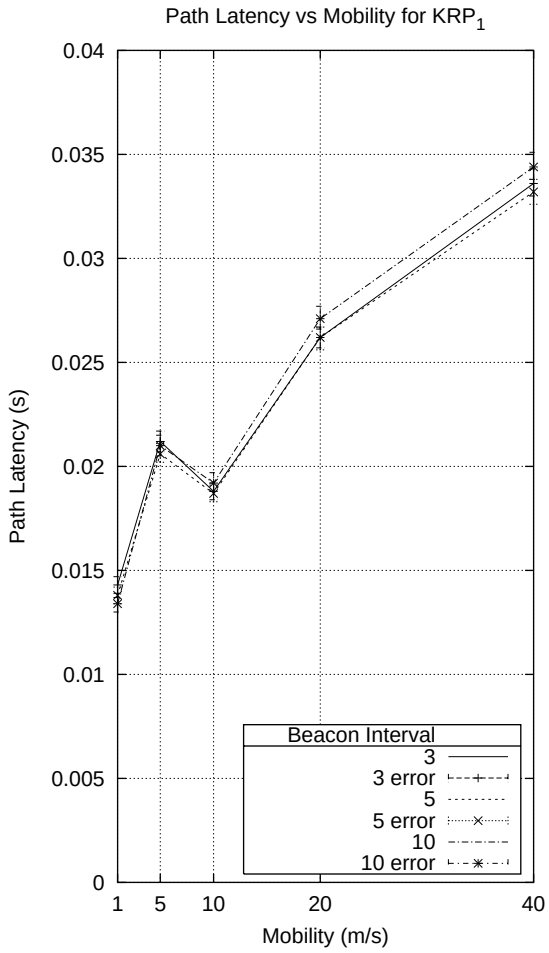

Figure 5.16 End-to-end Packet Latency for KRP_1 with 10 connections.

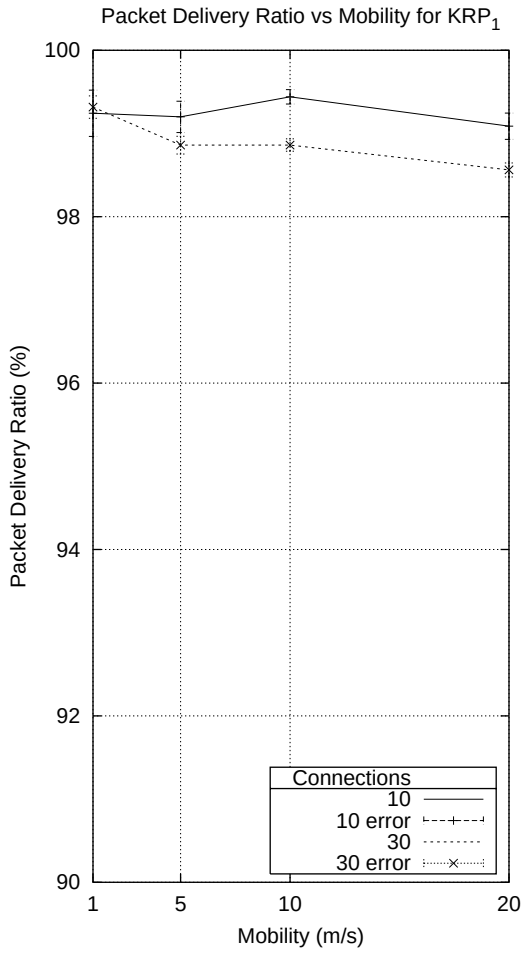

Figure 5.17 Packet Delivery Ratio for KRP_1 with beacon interval of 3.

Figures 5.17, 5.18, and 5.19 compare KRP_1 with 10 and 30 connections against mobility rates up to a maximum speed of 20 m/s. The beacon interval for KRP_1 is 3 for both traffic amounts in the graph.

With fewer connections, the performance of KRP_1 shows significant improvement, as indicated in Figure 5.17. The reduced contention is largely responsible for this change, and can be supported by looking at the graph of control overhead in Figure 5.18. The figure reveals that tripling the number of connections results in a more than 6-fold increase in overhead. The conclusion here is that the beaconing mechanism does not scale well as the number of connections increases.

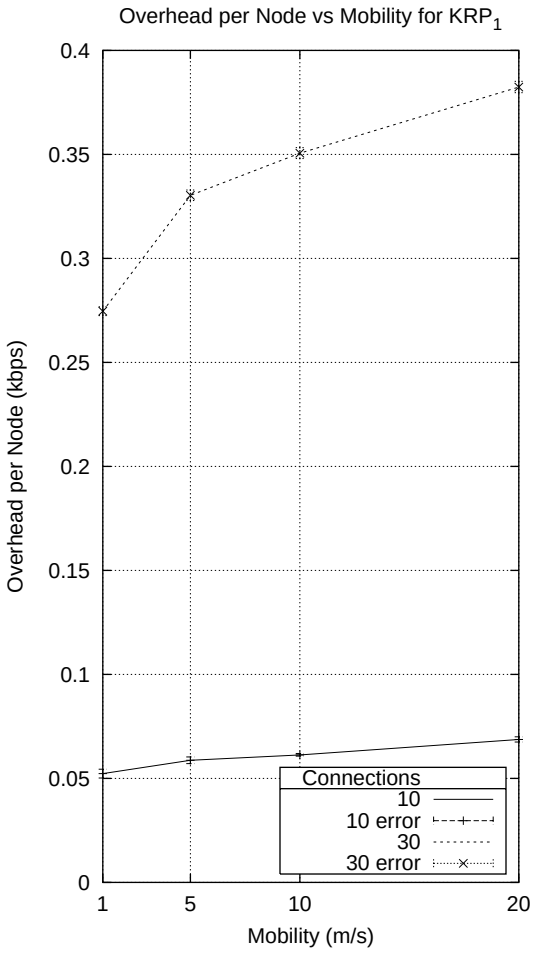

Figure 5.18 Control Overhead for KRP_1 with beacon interval of 3.

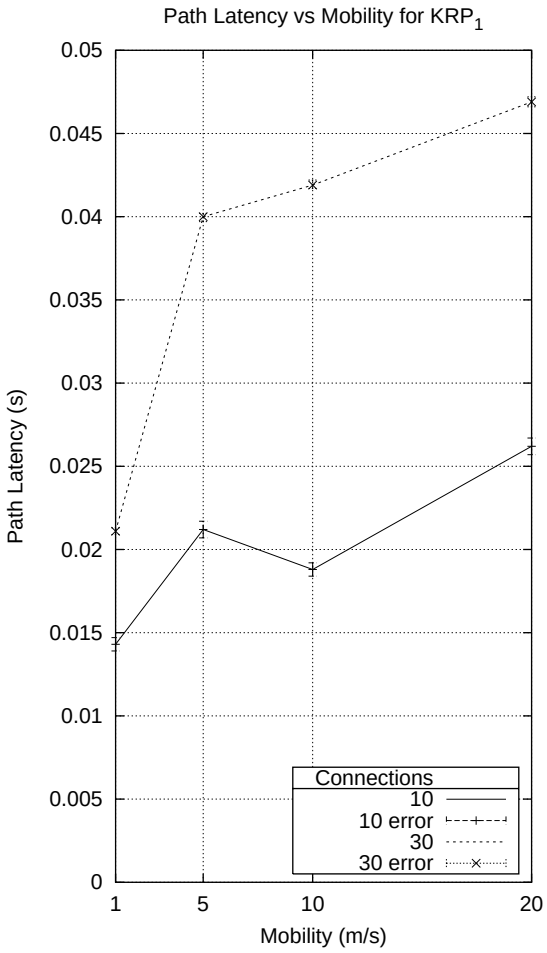

Figure 5.19 End-to-end Packet Latency for KRP_1 with beacon interval of 3.

Figure 5.19 confirms that the latency is considerably less with fewer traffic sources.

The trace files uncovered a problem in the beaconing mechanism that can adversely affect the setup of a remote connection between two MHs. Recall that a MH will respond to a BEACON under the three conditions of Section 4.5.1, and previously expired entries for a MH in the BB are erased when a GW discovers a new path to it. Under these rules, the following situation of Figure 5.20 can occur when a GW forwards its first data packets to a MH. In the figure, MH **A** (the destination) is within range of GW **G1**, and earlier responded to the first BEACON it heard from that GW. However, the BB entry for this path has since expired and was removed when GW **G2** received a reply from **A** through

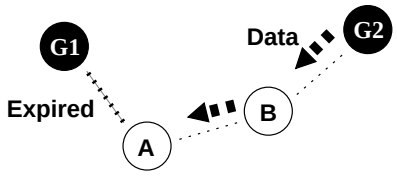

Figure 5.20 A problem with initial setup of a connection in KRP_1.

MH **B** in regards to a recent BEACON. As is apparent, the BB at this point does not reflect the shortest path to **A**. Hence, the first data packets arrive at **G2** for **A** based on the information in the BB. The situation is not rectified until either 1) the path between **G2** and **A** breaks and the recovery process culminates in a page for **A** in which case **G1** will again pick up a path to **A**, or 2) **G1** sends its next BEACON and **A** responds due to condition 3 of Section 4.5.1. However, the second case does not *always* fix the situation, as described below. Although this event can lead to a non-optimal start for a connection, we do not offer any optimization for it as it is only a transitory condition, and our connections are long-lived. When considering shorter lived connections, a solution to this temporary inefficiency is necessary.

A second problem that exists in KRP_1 is based on the order of steps taken by a GW to determine the next-hop for a destination when forwarding packets. In the implementation used for our experiments, a GW always prefers a local path to a MH over that of forwarding packets to another GW. In other words, a GW first looks in its own local routing table for a path to the destination before consulting the BB. While there may be advantages for this order of path checking depending on the design of the network, in our simulations it often leads to non-optimal path lengths. For example, if the source and destination of a connection are within local scope of the same GW (**G1**) and are both two hops away from it, the resulting path length is 4. If a different GW (**G2**) is only one hop away from the destination, the path length could be shortened to 3 hops, but this path will not be utilized due to **G1**'s bias. Decreasing the beacon interval can actually improve on this situation when the longer path to the destination breaks. After

the failure, the shorter path listed in the BB will be employed, potentially up until a reply is received by **G1** from the destination in response to its next BEACON packet. An enhanced version of KRP_1, denoted as KRP_1^E always prefers shortest paths. This version of the protocol also includes two other optimizations, described below.

In the experiments of Section 5.5, KRP_1 is diligent about re-establishing connections after experiencing a breakage. What this means is that the parameters `DISCOVERY_ATTEMPTS` and `PAGE_ATTEMPTS` are both set to a high value, in our case 32. The first few attempts follow an exponential back-off scheme for each successive failure, after which an attempt will be made roughly every second. For a source or destination that moves outside the bounds of the k-hop network, an exceptional amount of overhead can be produced in the search process. Moreover, the search process repeats itself after the last attempt. This led us to incorporate two new features into the protocol, resulting in KRP_1^E. The first is a GW-maintained list of destinations that are deemed "unreachable", after a limited number of page attempts (`PAGE_ATTEMPTS` is set to 3 in KRP_1^E) fails to produce a path. A GW that receives a data packet for a destination entered on the unreachable list are simply dropped. A MH is removed from the list when it responds to a BEACON and is entered in the BB. The second optimization is the creation of a new state that a MH enters after failing to find any GWs. This state, called the *WAIT* state, is triggered after just 3 failures, rather than 32. Once in the WAIT state, a MH will drop all outgoing data packets. The state is exited upon hearing a new BEACON from a nearby GW.

To test the performance of KRP_1^E, we used it in the same experiment that produced Figures 5.15 through 5.16. Figures 5.21 through 5.23 compare KRP_1 and KRP_1^E with a beacon interval of 3 for the three metrics.

KRP_1 demonstrates a higher packet delivery ratio over that of KRP_1^E in Figure 5.21 primarily due to the second and third optimizations of KRP_1^E, which limit the length of time MHs and GWs spend searching for an unreachable destination. Chapter 7 goes into more detail about the trade-off of these two optimizations.

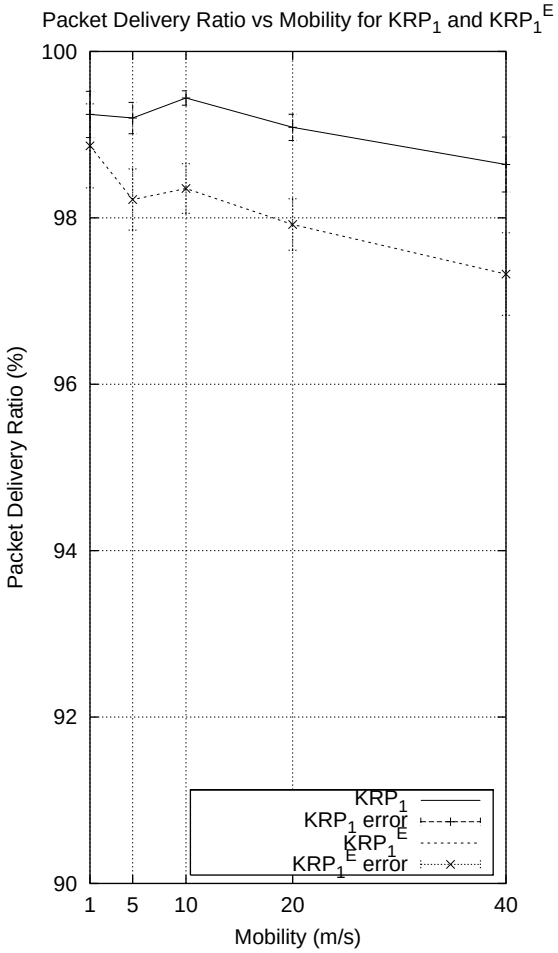

Figure 5.21 Packet Delivery Ratio for KRP_1 and KRP_1^E with 10 connections.

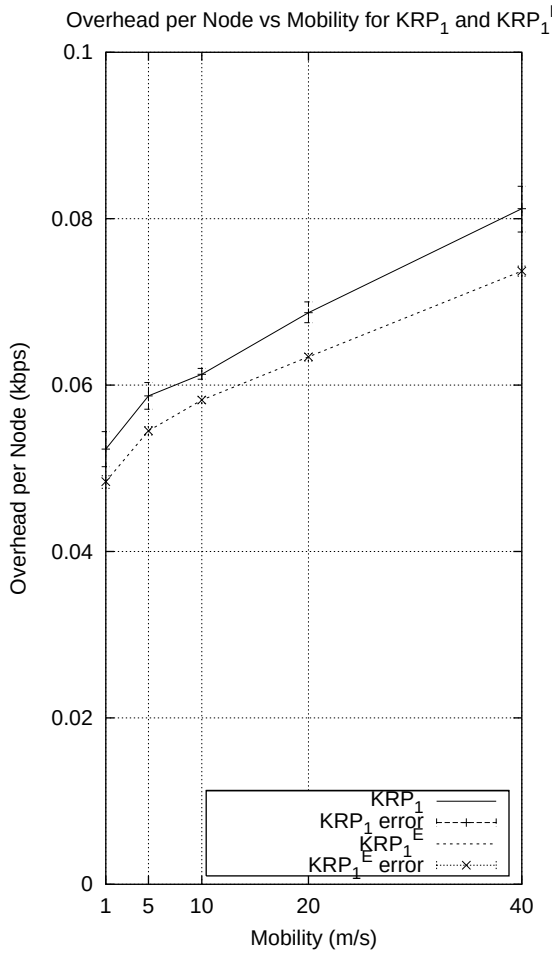

Figure 5.22 Control Overhead for KRP_1 and KRP_1^E with 10 connections.

Figure 5.22 shows that KRP_1^E requires less overhead, as a result of the three optimizations combined. However, a sacrifice is made in terms of packet delivery ratio, as previously noted. One could argue though that as the number of unreachable destinations increases (by decreasing k, for example), the second and third optimizations are necessary, especially when the number of connections is high.

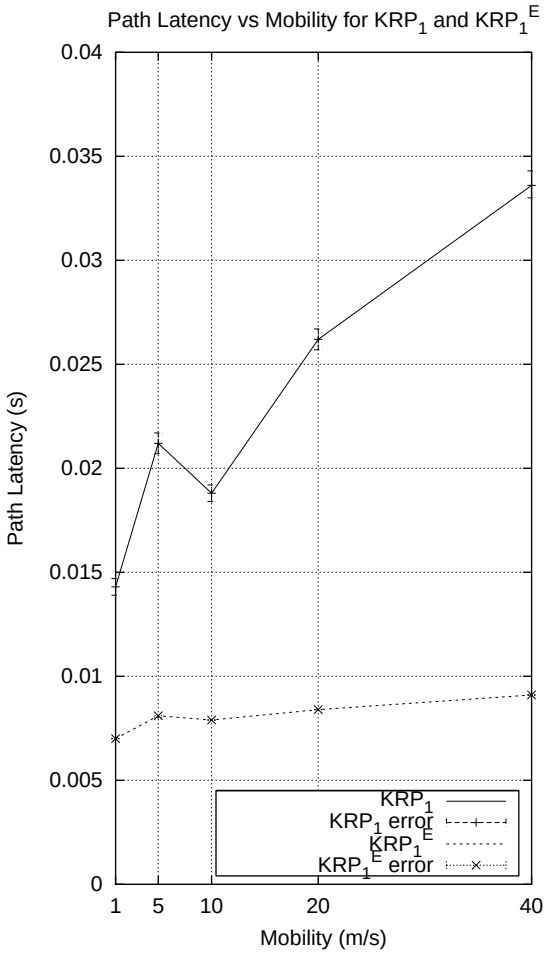

Figure 5.23 End-to-end Packet Latency for KRP_1 and KRP_1^E with 10 connections.

Quite an improvement in path latency is made for KRP_1^E versus KRP_1 as shown in Figure 5.23. The first optimization helps to reduce the number of hops required for an average connection. KRP_1's continual searching for unreachable destinations also partially contributes to its longer path latencies, slowing down data traffic. Moreover, packets can spend much longer time periods in a packet queue before being sent or forwarded in KRP_1 (maximum of 23 seconds recorded at 20 m/s), versus in KRP_1^E (maximum of 0.04 seconds recorded).

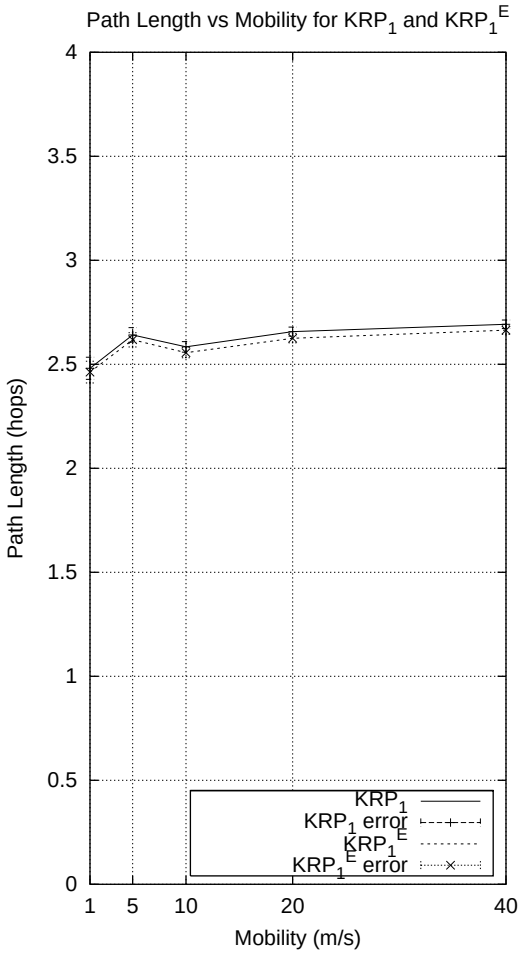

Figure 5.24 Average Path Length for KRP_1 and KRP_1^E with 10 connections.

Finally, Figure 5.24 further supports the benefit of the first optimization, as the average path length of KRP_1^E is indeed less than that of KRP_1 across all mobility speeds.

Yet another explanation for the seemingly backward trends of Section 5.5 stems from the heavy use of a few MHs for relaying packets by those MHs that are more than one hop away from a GW. The beaconing process results in a graph that looks like a tree with expanding branches, as illustrated in Figure 5.25. The constrained locality of packet transmissions may adversely affect the networks performance. However, when a MH's path to the GW breaks, it performs a search for `ALL_GWS`, and is more than likely to establish a path through a different MH that is less busy. Thus, decreasing the beacon

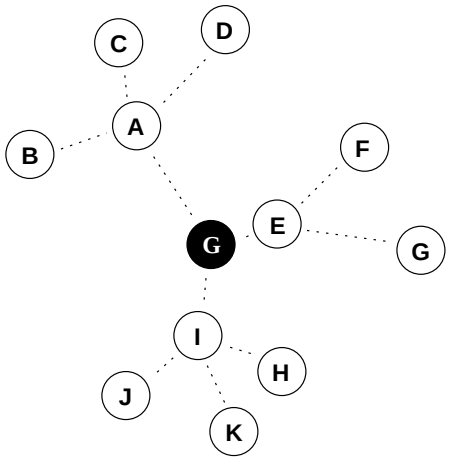

Figure 5.25 Tree formation from the beaconing process in KRP_1.

period allows these "faster" connections to remain usable for longer periods of time, which can improve the packet delivery ratio and latency metrics.

We may begin to realize at this point that the performance of KRP_1 is highly sensitive to the various parameters. Another parameter that has not received much attention is the maximum length of time a MH will defer when forwarding a broadcast packet. Recall from Section 4.5.1 that this value is set to 5 ms by default. Although we do not discuss this parameter in detail, preliminary tests over a range of defer values indicates that much lower packet latencies can be achieved when the range is increased. The effect this has on the network is to stretch the beaconing cycle over a longer period of time, reducing the amount of contention and allowing more packets to be retransmitted fewer times. This is crucial in terms of broadcast packets, which are never retransmitted. As in most routing protocols, control packets that are lost in KRP_1 can greatly reduce the network's ability to route packets successfully. Increasing the broadcast defer interval appears to improve control packet delivery percentages, thus leading to an overall improvement in the network performance.

CHAPTER 6

TESTBED

A testbed demonstrating the k-hop network architecture has been in place for just under a year in the CSL[1] building. It currently consists of nine Dell laptops equipped with Cisco Aironet 350 wireless Ethernet cards and Redhat Linux 7.3. The Aironet cards operate in ad hoc mode at all times. It is important to note that the building is equipped with several WLAN APs[2] that add some contention to the channel. The beacons from the APs are ignored by the Aironet cards while in ad hoc mode, and do not join in the network. In our testbed, any laptop housing a wired Ethernet interface in addition to the Aironet card can operate as a GW. An Ethernet LAN serves as the backbone network for the GWs.

Over the past year, we have implemented routing protocols emulating the behavior of KRP_1 and KRP_2, and have also implemented a simple reactive protocol similar to standard AODV [14]. However, there are differences between the testbed and simulation versions for each protocol, which are mentioned below.

KRP_1 in our testbed represents an early version of the one simulated in this thesis. The details of the protocol can be found in [65]. BEACON packets are periodically issued by GWs, and MHs hearing a beacon respond with a JOIN message (essentially a RREP). However, in our testbed protocol, a MH will only join *one* GW at a time, whereas in our

[1] Coordinated Science Laboratory
[2] Access Points

simulated protocol, a MH will respond to multiple GWs. Moreover, the MH will respond to *every* BEACON from its registered GW; in KRP_1 of Section 4.5, a MH only responds under the three conditions listed. The beaconing process is also extended to include a unicast acknowledgment (JOIN-ACK) from the GW to the MH in response to receiving a JOIN message. If the MH does not receive a JOIN-ACK from the GW it erases the route. This added procedure helps to confirm paths that are suitable for data transfers, but at the same time invites more control overhead into the system. A second major difference between the two protocols is default routes in the testbed version of KRP_1 are not present, whereas in the simulation version they are heavily used. Instead, MHs in the testbed are always required to initiate a RREQ to a destination that is not listed in its routing table, and GWs answer on behalf of the destination when receiving a RREQ. The RREP in this case includes the *total* hop count of the path, i.e., the number of hops on both sides of the GW from the source to the destination. MHs distinguish between local and remote paths by setting a flag in the routing table entry for connections that pass through a GW. This allows for a path to be chosen (in the case where more than one path is returned) based on this property.

KRP_2 was most recently added to our testbed, and closely follows the simulation edition. A noticeable difference is that KRP_2 as presented in Section 4.6 temporarily buffers data packets when a path is unavailable to a given destination, while in the testbed, the packets are simply dropped. Depending on the traffic type, it can be argued that one or the other approach is desirable. For example, for voice traffic it might be better to simply drop the packets, since they will not meet their deadlines. Conversely, for TCP traffic, some delay is tolerable.

The routing protocol suite of the testbed includes added features that are not necessary for simulation. First, the rollover of sequence numbers is supported. Second, startup after a MH or GW crash is handled by storing to disk the latest sequence number of the source after each increment. A MH or GW that re-enters the network must make sure that its sequence number will be "higher" than anyone else's sequence number for itself.

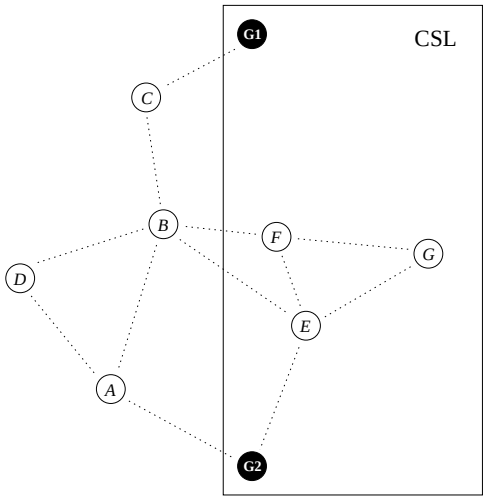

Figure 6.1 Testbed setup with two GWs and seven MHs.

Alternatively, it can wait after for a designated period of time before rejoining, ensuring that the rest of the network has "forgotten" about it (i.e., all paths to the MH or GW have expired and been deleted).

In a real network, there are many more concerns beyond the routing protocol. One of these concerns involves dynamic address assignment. To solve this problem, we have extended the operation of DHCP[3] [66] through the use of relays. Another concern is distinguishing between ad hoc (internal) and external hosts. In our testbed, all addresses offered by the DHCP server contain the subnet mask 192.168.0 (not addressable from the Internet), allowing GWs and mobile hosts to make smart forwarding decisions based on the destination address.

Figure 6.1 illustrates one sample configuration for the testbed. Nodes **A**, **B**, **C** and **D** are placed in the field and communicate with GWs **G1** and **G2** to the remaining nodes **E**, **F**, and **G** that are inside the building. Internet access is provided through **G1** and **G2**. **B**, **D**, **F**, and **G** require multiple hops to reach the GWs.

[3]Dynamic Host Configuration Protocol

We believe the k-hop paradigm to be especially well-suited for small, busy geographic areas, such as a university campus or airport. For example, there may be GWs inside a campus building, but users just outside the building cannot access it. However, if there are other users, or dedicated relays, in the lobby of the building, such devices can serve as routers to extend the range of the GW to outside users. Building a testbed allows us to explore issues which may arise in implementation that are not handled in simulation. The system does not have to rely on a simulator's modeling of mobility, obstacles and fading. With this testbed, we plan to develop useful upper layer applications that take advantage of the architecture.

In the following sections, we explain in greater detail some of the more interesting aspects of our testbed.

6.1 Data Forwarding

Data forwarding is performed by the kernel, and is independent of the routing protocol used. There are only two cases in which the two entities interact — 1) A data packet with an unknown destination arrives at the kernel, captured using the technique described in Section 6.3, or 2) a link-layer failure is detected via the method of Section 6.4. Since the forwarding table specifies an interface, there is no special handling for GWs. Each forwarding entry expires after a certain amount of time unless refreshed. Entries are refreshed after being used to forward data.

External hosts (i.e., a host outside of the ad hoc network) are informed of any undeliverable data packets that arrive at a GW using an ICMP [67] unreachable message. Note that if the ad hoc nodes are given globally unroutable IP addresses via DHCP, as is the case in our testbed, a GW should not expect to receive any incoming packets for which it does not have an associated connection already established (GWs employ IP masquerading to provide Internet connectivity to the MHs). Outgoing data packets that are destined for an external host in our testbed are forwarded using the default route at

each hop. If the host does not exist or is not reachable (after passing through one of our GWs), an ICMP destination unreachable message is returned.

6.2 Address Assignment

For most networks, a node should be allowed to enter and leave at will, which requires dynamically assigned addresses. Automatic address assignment in IPv4 networks is often handled by DHCP [66]. The DHCP protocol requires that each host be within one hop of either a DHCP server or a DHCP relay. Relays are configured with the IP address of the server, allowing it to be several hops away with routers in between. In networks with ad hoc nodes, neither the server nor the relay are likely to be within one hop of the node requesting configuration. Thus, the protocol has been viewed as a poor choice for ad hoc networks with multiple hops.

There are many proposed schemes for dynamic address assignment in ad hoc networks [68, 69], along with methods for detecting duplicate addresses when partitions merge [70]. Although each scheme has its advantages, we opted to stick with DHCP given its proven track record and popularity with existing networks, along with the fact that the k-hop network architecture requires each node to be within reach of the fixed infrastructure, and thus has access to the server.

Address assignment to nodes that are multiple hops away from a GW is achieved by requiring all addressed nodes to act as a DHCP relay. The actual steps used for obtaining an IP address are identical to those used in the DHCP standard. In short, for an unassigned node **N**, the following events are performed:

1. **N** broadcasts a DHCP DISCOVER message
2. Neighboring relays of **N** forward the DISCOVER message to the DHCP server.
3. A DHCP OFFER message is generated by the server and unicast back to **N** via the relays.

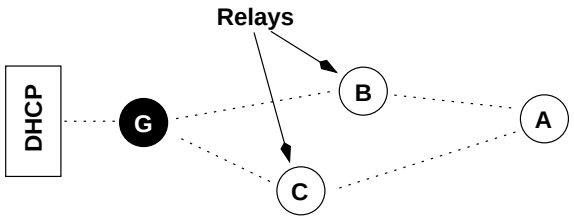

Figure 6.2 A problem with DHCP in ad hoc networks.

4. **N** and the DHCP server exchange one or more DHCP REQUEST/OFFER messages followed by a DHCP ACK message from the server.

5. **N** becomes a DHCP relay to help other nodes farther away obtain IP addresses.

The DHCP relay daemon must be supplied the IP address of the DHCP server. This address can be obtained from the "server identifier" option field of the DHCP packets. Once the daemon has been started, it listens for DHCP packets and forwards them toward the DHCP server. The underlying routing protocol automatically handles the forwarding process.

A drawback of the above technique is the unnecessary forwarding by multiple relays if the network is dense. For example, in Figure 6.2, node **A** has two neighboring relays, nodes **B** and **C**. Assuming that **A** has just joined the network, to obtain an address it broadcasts a DHCP DISCOVER message, which is heard by **B** and **C**. In this case, both nodes will forward the message to GW **G**, when in fact only one node needs to do so. Redundant requests received by the DHCP server will result in extra DHCP OFFERS. One way to overcome this situation (although not implemented in the testbed) is to allow nodes to monitor the channel and suppress its own transmission if it overhears a neighbor forwarding the same message. For more techniques for address assignment using DHCP, see [65].

6.3 Capturing Unroutable Packets

In our testbed, the responsibility of capturing outgoing data packets that do not have a forwarding entry in the kernel rests on the ASL[4] [71], an API library designed to help in the development of ad hoc routing protocols. The ASL accomplishes this feat by establishing a "catch all" entry in the forwarding table which leads to a virtual tunneling device (TUN/TAP). Packets matching this mask (or everything if set as the default route) are passed to this device, transferring it from kernel space to user space. The packet can then be processed by the routing protocol, and the appropriate actions taken, such as route discovery or issuing an error. Through simple function calls, the ASL can be instructed to hold additional data packets while route discovery is being performed, and released when discovery is completed. If the destination is found, the packets are sent via a raw socket to avoid duplicate IP headers. The ASL also includes a module which records how recently forwarding entries are used. By using the ASL, we overcome the large overhead of copying every data packet back and forth between user and kernel space, a limitation seen in other ad hoc implementations.

6.4 Detecting Link Breakages with 802.11

The routing protocols in our testbed rely on feedback from the link-layer to detect and handle link breakages. Until recently, there did not exist any widely available method for obtaining this information from the Aironet cards. It is now accessible using Wireless Tools [72] for Linux, an API that serves as a bridge between device drivers and user space programs. While the mechanism the tools provide for identifying link failure is crude, it enables the routing protocols to run more efficiently. The API at this point in time is unable to distinguish the source of a dropped packet, a limitation that restricts the testbed's ability to efficiently report link failures to other nodes in some cases.

[4] Ad Hoc Support Library

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The purpose of this thesis was to develop a set of routing protocols that are tailored for the architecture described in Chapter 3. This process culminated in the creation of two protocols, KRP_1 and KRP_2. The performance of KRP_1 has been shown through simulation to have a constant advantage over that of KRP_2. However, for low mobility rates and k no greater than 2, KRP_2 is a suitable protocol, offering lower latency than KRP_1 in most cases. Perhaps most important are the following observations about each protocol's behavior that can help to improve their performance and serve as key points of consideration in developing new protocols.

One issue that was not immediately addressed in either protocol is that of efficient handling for MHs that enter and leave the network frequently. For example, a MH participating in a connection that loses contact with all of the GWs in its GL becomes severed from the network, and attempts to re-establish the connection by searching for new GWs. If disconnected long enough, the MH will cease the search process and will subsequently wait until it hears from a GW. Due to the periodic nature of both BEACONS (KRP_1) and update packets (KRP_2), there often exists a time between which the MH *is able* to resume the connection and when it actually does resume. This interval becomes longer when the frequency of BEACONS or updates decreases. Depending on the type of application, this delay may or may not be acceptable. However, a balance must be struck between the amount of overhead required for the search process and the latency incurred

from not searching. In our simulations, packets are generated by the source regardless of whether it is connected, resulting in large packet drops for periods of time when the MH is disconnected; in practice a MH that has discovered that it is disconnected would most likely halt or buffer outgoing packets.

As is evidenced from our simulations, KRP_1 does not seem to benefit significantly from the added beaconing feature as originally anticipated. The purpose of the beaconing is to update routes from MHs to their nearby GWs pro-actively in the hopes of mitigating future link breakages. However, link breakages still occur even with this mechanism, thus comprising its overall effect. A disadvantage of the technique in addition to that just stated derives from its own periodic nature. Following a BEACON, MHs are required to respond under the set of conditions mentioned in Section 4.5.1. In cases where the network is sufficiently dense and many connections exist or when the mobility rate is high (or both), the flood of responses causes brief moments of high contention in the network, leading to high packet delays and packet drops. This situation is more pronounced as k increases. Even though each responding MH waits a random amount of time before sending its reply, an approach that is more distributed over time may be favorable to fixing this problem. One such solution is proposed next, and is left as a future experiment.

To reduce the number of transmissions and subsequently the amount of contention in the network, the response phase could follow a simple deadline model. This model serves to aggregate responses as they move toward the GW. Particularly when k is larger than 2 or 3, aggregation becomes necessary in order to keep overhead costs down. The technique works as follows. Upon forwarding a new BEACON, each MH sets a timer to expire after $\frac{k^2-kh}{\tau}$ s, where h represents the number of hops that the beacon has traveled thus far, and τ is the beacon interval. The timer indicates a deadline by which the MH must send its own response. The resulting range for deadlines is $[0, \frac{k^2+k}{\tau}]$. For MHs that are on the perimeter of the network, it is not necessary to defer responses, and hence the deadline is 0. However, MHs that are within direct range of the GW must send its aggregated response by time $t + \frac{k^2+k}{\tau}$, where t is the time that the beacon was issued from the

GW. Any responses that are received by a MH before its deadline expires are included in that MH's reply. Responses that are heard after the deadline are either dropped or forwarded independently. It is important to note that a certain amount of jitter must also be incorporated into the scheduling to avoid contention at each deadline.

Another conclusion is that the semantics of KRP_2 does not lend itself well to a multi-path scheme. More investigation is necessary to improve the average lifetime of the multiple paths. Figure 7.1 demonstrates the drawback of asynchronous updates. In Figure 7.1a, MH **A** receives an update from MH **B** regarding a route to MH **C**. The update specifies a new sequence number for **C**, so **A** updates its route to **C** by deleting the existing paths in the routing table and adding the new one via **B**. Next, in Figure 7.1b, the link between **B** and **C** breaks, and an update is issued by **B** and is overheard by **A**. **A** promptly invalidates its route to **C** by incrementing the sequence number by one. Shortly after, in Figure 7.1c, **A** receives an update from MH **D** offering a route to **C** in two hops, but at this point it is too late, since the sequence number for **C** in **D**'s update is now old. However, the route is valid and should be accepted. This situation is much less frequent in KRP_1 due to the fact that all new paths are learned within a very short period of time. However, even if a solution is devised to do away with this shortcoming, the next question to address is how **A** should react to receiving the invalidation update from **B** followed shortly by the valid update from **D**. If **A** issues its own update in response to both events, a considerable amount of additional control traffic may be introduced in the network. If not, other MHs might be forced to find a different path to **C** that does not include **A**. One approach is to keep track of upstream MHs that are concerned about the route to **C** (or any other MH reachable through **C**), similar to AODV's precursor lists [14]. This way, **A** can suppress sending an invalidation update if no precursors exist. The precursor list in this approach would be populated by observing the source and destination addresses of data packets.

An approach not implemented in this thesis involves modifying DSR [11] to fit the k-hop architecture. The necessary changes to the basic protocol would be nearly identical to

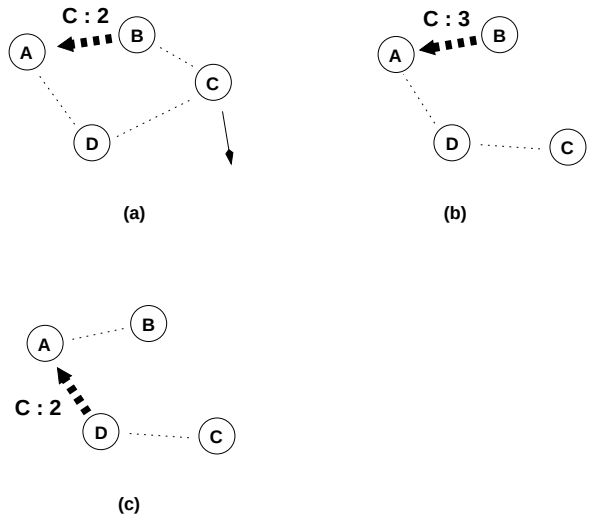

Figure 7.1 Paths lost due to the nature of KRP_2's sequence numbers.

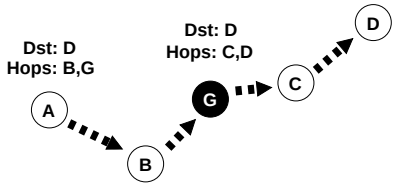

Figure 7.2 Source routing with default routes.

those applied to KRP_1. The main advantage of using DSR as the underlying protocol is that the protocol is well-suited to support multiple path routing. In addition, since the number of wireless hops is limited to k, the extra overhead caused by source routing is bounded. In fact, both KRP_1 and KRP_2 possess nearly the same amount of overhead as source routing due to the requirement of IP-in-IP for data packets that are forwarded to a GW. The extra overhead incurred by source routing can be further reduced by only including *half* of the source route when using GWs. For example, in Figure 7.2, a data packet from MH **A** destined for MH **D** needs only to contain MH **B** and GW **G** in the packet header until it reaches **G**, after which the header is modified to contain **C** and **D** as the route to use. We plan to explore this alternative in the future.

An area of future work beyond this thesis is the introduction of fixed *relays* that provide connectivity support for sparsely populated networks. Relays are positioned strategically throughout the network and serve as forwarding points for MHs that are more than one hop away from the GW. Use of relays also takes the burden of packet forwarding off of intermediate MHs, where conservation of resources such as battery power might be of importance. Fixed relays (or mobile relays that are able to identify their own location) serve as an extension of the backbone infrastructure, and hence can employ more efficient routing techniques and provide a more accurate depiction of each MHs location in network.

Another topic which deserves some attention in the future is that of congestion avoidance. It is common in ad hoc networks that the shortest path found is not necessarily the most reliable [73]. This can be due to several factors such as high interference on a link, or high contention around a particular part of the network. These unreliable links result in many unnecessary retransmissions. However, longer links are also not attractive due to self-interference [3]. Therefore, predicting efficient routes is a daunting task. Recent studies of existing public-area wireless networks indicates that user load is often distributed unevenly among wireless access points (APs) [74, 75, 76]. To avoid this situation, we suggest a simple congestion control algorithm to help balance the load more evenly among the GWs. The scheme is similar to the hot-spot avoidance techniques noted in Section 2.

To estimate the amount of traffic flowing through itself, a GW records the size and number of packets that it either forwards or sends. At any given instant of time, the product of these two values represents the instantaneous load of the GW. Averaged over a longer time span, this value serves as a load indicator for the GW. MHs within reach of multiple GWs use the load indicator (LI) to make data forwarding decisions. Thus, the LI is included in the header of packets originating from the GW. A GW that has a very low LI and is 3 hops away will be chosen over a GW that is only 2 hops away, but has a much higher LI. The congestion control algorithm therefore takes into account not only

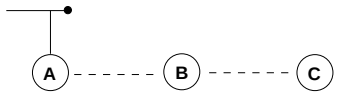

Figure 7.3 A sample ad hoc network with wireline drop.

the path length, but also the load experienced by the GWs. When making forwarding decisions, GWs prefer routes with the best path length/LI combination.

Finally, experiments with our testbed bring up several interesting issues that must be addressed. One such issue concerns setting up socket connections in an ad hoc network with programs such as SSH[1]. Consider the topology depicted in Figure 7.3, where 3 MHs are arranged side-by-side with MH **A** possessing a wireline interface in addition to to its wireless one. None of the MHs are acting as a GW in this example. Routing protocol communications easily set up the routing tables at all three hosts for packet forwarding. However, when attempting to initiate a SSH session between MHs **A** and **C**, a problem occurs. Specifically, when **A** constructs its first TCP packet, the application places the *wireline* address of **A** in the source address of the IP header, which is then forwarded to **C**. Upon receiving this initial packet, **C** creates a new socket to the address specified in the packet, which, in this case, is not routable. As a result, the connection cannot be properly established. This problem is due to a lack of control over IP address selection when creating connections. Ideally, **A** would take into account the destination's address before deciding which interface address is the most appropriate choice.

[1]Secure Shell

REFERENCES

[1] Charles E. Perkins, editor. *Ad Hoc Networking*, chapter 1, pages 8–14. Addison Wesley, 2001.

[2] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[3] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, July 2001.

[4] Michael Gastpar and Martin Vetterli. On the capacity of wireless networks: The relay case. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, June 2002.

[5] César A. Santiváñez, Bruce McDonald, Ionnis Stavrakakis, and Ram Ramanathan. On the scalability of ad hoc routing protocols. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, June 2002.

[6] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Dallas, TX, October 1998.

[7] Charles E. Perkins, Elizabeth M. Royer, Samir R. Das, and Mahesh K. Marina. Performance comparison of two on-demand routing protocols for ad hoc networks. *IEEE Personal Communications*, 8(1):16–28, February 2001.

[8] Atsushi Iwata, Ching-Chuan Chiang, Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1369–79, August 1999.

[9] César A. Santiváñez, Ram Ramanathan, and Ionnis Stavrakakis. Making link-state routing scale for ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Long Beach, CA, October 2001.

[10] Christian Tschudin and Richard Gold. LUNAR: Lightweight underlay network ad-hoc routing. Technical report, University of Basel, Switzerland, January 2002.

[11] David B. Johnson and D.A. Maltz. Dynamic source routing in ad hoc wireless networks. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[12] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks (DSR). IETF Internet Draft, *draft-ietf-manet-dsr-07.txt*, Work-in-progress, February 2002.

[13] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, New Orleans, LA, February 1999.

[14] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (AODV) routing. IETF Internet Draft, *draft-ietf-manet-aodv-12.txt*, Work-in-progress, November 2002.

[15] Vincent D. Park and M. Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Kobe, Japan, April 1997.

[16] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi. Signal stability-based adaptive routing (SSA) for ad-hoc mobile networks. *IEEE Personal Communications*, 4(1):36–45, February 1997.

[17] C-K. Toh. Associativity based routing for ad hoc mobile networks. *Wireless Personal Commmunications Journal, Special Issue on Mobile Networking and Computing Systems*, March 1997.

[18] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector (DSDV) routing for mobile computers. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, London, UK, August 1994.

[19] Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications, Special Issue on Routing in Mobile Communication Networks*, 1(2):183–97, October 1996.

[20] J. J. Garcia-Luna-Aceves and Marcelo Spohn. Efficient routing in packet-radio networks using link-state information. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, September 1999.

[21] Tsu-Wei Chen and Mario Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, Atlanta, GA, June 1998.

[22] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing: A new routing scheme for ad-hoc wireless networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, New Orleans, LA, June 2000.

[23] Mario Gerla, Xiaoyan Hong, and Guangyu Pei. Fisheye state routing protocol (FSR) for ad hoc networks. Internet Draft, *draft-ietf-manet-fsr-03.txt*, Work-in-progress, June 2002.

[24] Philippe Jacquet and Paul Mühlenthaler. Optimized link state routing protocol for ad hoc networks. In *Proceedings of the IEEE International Multi-Topic Conference (INMIC)*, Pakistan, June 2001.

[25] Philippe Jacquet, Paul Mühlenthaler, and Amir Qayyum. Optimized link state routing protocol. Internet Draft, *draft-ietf-manet-olsr-07.txt*, Work-in-progress, December 2002.

[26] Bhargav Bellur and Richard G. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, March 1999.

[27] Richard G. Ogier, Mark G. Lewis, Fed L. Templin, and Bhargav Bellur. Topology dissemination based on reverse-path forwarding (TBRPF). IETF Internet Draft, *draft-ietf-manet-tbrpf-06.txt*, Work-in-progress, November 2002.

[28] Zygmunt J. Haas and Marc R. Pearlman. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC)*, San Diego, October 1997.

[29] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The zone routing protocol (ZRP) for ad hoc networks. Internet Draft, *draft-ietf-manet-zone-zrp-04.txt*, Work-in-progress, July 2002.

[30] Kaixin Xu, Xiaoyan Hong, and Mario Gerla. An ad hoc network with mobile backbones. In *Proceedings of the IEEE International Conference on Communications (ICC)*, New York, NY, April 2002.

[31] Kaixin Xu and Mario Gerla. A heterogeneous routing protocol based on a new stable clustering scheme. In *Proceedings of the Military Communications Conference (MILCOM)*, Anaheim, CA, October 2002.

[32] Daniel Lihui Gu, Gaungyu Pei, Henry Ly, Mario Gerla, and Xiaoyan Hong. Hierarchical routing for multi-layer ad-hoc wireless networks with UAVs. In *Proceedings of the Military Communications Conference (MILCOM)*, Anaheim, CA, October 2002.

[33] A. Bruce McDonald and Taieb F. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1466–87, August 1999.

[34] Stefano Basagni, Damla Turgut, and Sajal K. Das. Mobility-adaptive protocols for managing large ad hoc networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, Helsinki, Finland, June 2001.

[35] Ram Ramanathan and Martha Steenstrup. Hierarchically-organized, multihop mobile wireless networks for quality-of-service support. *Mobile Networks and Applications*, 3(1):101–19, 1998.

[36] Mario Joa-Ng and I-Tai Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1415–25, August 1999.

[37] Ching-Chuan Chiang and Mario Gerla. Routing and multicast in multihop, mobile wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC)*, San Diego, October 1997.

[38] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang. A wireless hierarchical routing protocol with group mobility. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, September 1999.

[39] Mario Gerla, Xiaoyan Hong, Li Ma, and Guangyu Pei. Landmark routing protocol (LANMAR) for large scale ad hoc networks. IETF Internet Draft, *draft-ietf-manet-lanmar-05.txt*, Work-in-progress, November 2002.

[40] Suman Banerjeea and Samir Khuller. A clustering scheme for hierarchical control in multi-hop wireless networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Anchorage, AK, April 2001.

[41] Chunhung Richard Lin and Mario Gerla. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–75, September 1997.

[42] P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. *ACM Computer Communication Review*, 27(2), April 1997.

[43] Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, mobile, multimedia radio network. *Wireless Networks*, 1:255–265, October 1995.

[44] Guangyu Pei, Mario Gerla, and Xiaoyan Hong. LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Boston, MA, August 2000.

[45] Bevan Das, Raghupathy Sivakumar, and Vaduvur Bharghavan. Routing in ad hoc networks using a spine. In *Proceedings of the IEEE International Conference on Computer Communications and Networks (IC3N)*, Las Vegas, NV, September 1997.

[46] Xiaoyan Hong, Mario Gerla, Yunjung Yi, Kaixin Xu, and Taek Jin Kwon. Scalable ad hoc routing in large, dense wireless networks using clustering and landmarks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, New York, NY, April 2002.

[47] Elizabeth Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, April 1999.

[48] Xiaoyan Hong, Kaixin Xu, and Mario Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network*, pages 11–21, July 2002.

[49] G. Aggelou and R. Tafazolli. On the relaying capacity of next-generation GSM cellular networks. *IEEE Personal Communications*, 8(1):40–47, February 2001.

[50] Y. Lin and Y. Hsu. Multihop cellular: A new architecture for wireless communications. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Tel Aviv, Israel, March 2000.

[51] X. Wu, S. H. Chan, and B. Mukherjee. MADF: A novel approach to add an ad-hoc overlay on a fixed cellular infrastructure. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Chicago, IL, September 2000.

[52] C. Qiao and H. Wu. iCAR: An intelligent cellular and ad-hoc relay system. In *Proceedings of the IEEE International Conference on Computer Communications and Networks (IC3N)*, Las Vegas, NV, October 2000.

[53] Anand Balachandran, Paramvir Bahl, and Geoffrey M. Voelker. Hot-spot congestion relief in public-area wireless networks. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, Callicoon, NY, June 2002.

[54] Jie Zhou and Yang Richard Yang. PARCelS: Pervasive ad-hoc relaying for cellular systems. In *Med-Hoc-Net'02*, Sardegna, Italy, September 2002.

[55] Hung-Yun Hsieh and Raghupathy Sivakumar. A hybrid network model for cellular wireless packet data networks. In *Proceedings of the IEEE Global Telecommunications Conference (Globecom)*, Taipei, Taiwan, November 2002.

[56] R. Sivakumar and H. Hsieh. On using the ad-hoc network model in wireless packet data networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[57] H. Hung-Yun and R. Sivakumar. Performance comparison of cellular and multi-hop wireless networks: A quantitative study. In *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, Cambridge, MA, June 2001.

[58] Hung-Yun Hsieh and Raghupathy Sivakumar. Towards a hybrid network model for wireless packet data networks. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, Taormina, Italy, July 2002.

[59] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. Intrazone routing protocol (IARP). Internet Draft, *draft-ietf-manet-zone-iarp-02.txt*, Work-in-progress, July 2002.

[60] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. Interzone routing protocol (IERP). Internet Draft, *draft-ietf-manet-zone-ierp-02.txt*, Work-in-progress, July 2002.

[61] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Seattle, WA, August 1999.

[62] The VINT Project. The network simulator — ns-2. `www.isi.edu/nsnam/ns`.

[63] The Monarch Project. Rice monarch project — mobile networking architectures. `www.monarch.cs.rice.edu`.

[64] IEEE Computer Society. *802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, June 1997.

[65] Matthew J. Miller, William D. List, and Nitin H. Vaidya. A hybrid network implementation to extend infrastructure reach. Technical report, University of Illinois, Champaign-Urbana, January 2003.

[66] Ralph Droms. Dynamic host configuration protocol. IETF Request for Comments (Standard) 2131, March 1997.

[67] Jonathan Postel. Internet control message protocol. IETF Request for Comments (Standard) 792, September 1981.

[68] Charles E. Perkins, Jari T. Malinen, Ryuji Wikikawa, Elizabeth M. Belding-Royer, and Yuan Sun. IP address autoconfiguration for ad hoc networks. IETF Internet Draft, *draft-ietf-manet-autoconf-01.txt*, Work-in-progress, November 2001.

[69] Sanket Nesargi and Ravi Prakash. MANETconf: Configuration of host in a mobile ad hoc network. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, June 2002.

[70] Nitin H. Vaidya. Weak duplicate address detection in mobile ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[71] Vikas Kawadia, Yongguang Zhang, and Binita Gupta. System services for implementing ad-hoc routing protocols. In *Proceedings of the International Workshop on Ad Hoc Networking (IWAHN)*, Vancouver, Canada, August 2002.

[72] Wireless Tools for Linux. `www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html`.

[73] Douglas S. J. De Couto, Daniel Aguayo, Benjamin A. Chambers, and Robert Morris. Performance of multihop wireless networks: Shortest path is not enough. In *Proceedings of the Workshop on Hot Topics in Networking (HotNets)*, Princeton, NJ, October 2002.

[74] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Atlanta, GA, September 2002.

[75] Diane Tang and Mary Baker. Analysis of a local-area wireless network. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, August 2000.

[76] Diane Tang and Mary Baker. Analysis of a metropolitan-area wireless network. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Seattle, WA, August 1999.