A Hybrid Network Implementation to Extend Infrastructure Reach

Matthew J. Miller William D. List Nitin H. Vaidya
University of Illinois at Urbana-Champaign
{mjmille2,list,nhv}@uiuc.edu

Technical Report
Initial Version: September 2002
Revised: January 2003

Abstract

This paper describes a hybrid network implementation that uses both ad hoc connectivity and access points. The network also allows mobile hosts that are multiple hops from an access point to use centralized services, like DHCP, which are not available in pure ad hoc networks. Many scenarios may benefit from this extension of services, such as mobile users near university buildings or at an airport. For efficiency, the "radius" of an access point is limited to K hops. This means all routes have at most K consecutive wireless hops before reaching the destination or an access point. We believe this limitation may lead to more efficient routing by trading off some connectivity. The protocol uses proactive routing at the access points and on-demand routing at the mobile hosts. We present an implementation done as proof-of-concept and a basis for future research.

1 Introduction

This report describes the implementation of a testbed for a network. A hybrid network is defined to be one that contains mobile devices, relay devices and access points. Mobile devices can be anything from laptops to PDA's to cell phones. Access points give mobile hosts (MHs) access to other devices outside of a MH's transmission range. An example of such a system is shown in Figure 1. In the figure, **MH2** can reach **MH5** despite the fact they cannot communicate in ad hoc mode. For the purposes of this paper, base station (BS), gateway (GW) and access point (AP) are synonymous. We also use the term mobile host (MH) and node interchangeably.

Current wireless infrastructure only provides services, such as DHCP and Internet connectivity, to MHs that are one hop away. That is, the MH is within transmission range of the AP and hence the two can directly communicate. This allows MHs to function like hosts on traditional shared media networks that access a gateway for outside connectivity and centralized services. However, this model does not take advantage of the plethora of research that has been done in ad hoc networks (discussed in Section 2) to allow MHs to communicate directly. By incorporating ad hoc networking into the existing infrastructure, the "range" of an AP can be extended to allow for more connectivity. For example, in Figure 1, the services of **BS2** can be extended to **MH6** and **MH7** as opposed to just **MH4** and **MH5**. Additionally, when ad hoc networking is incorporated, not all communication between MHs has to go through the AP. This may increase the spatial reuse of the channel and ease the burden placed on the AP. If **MH2** and **MH5** can communicate directly, we avoid using resources such as the base stations and wired network.

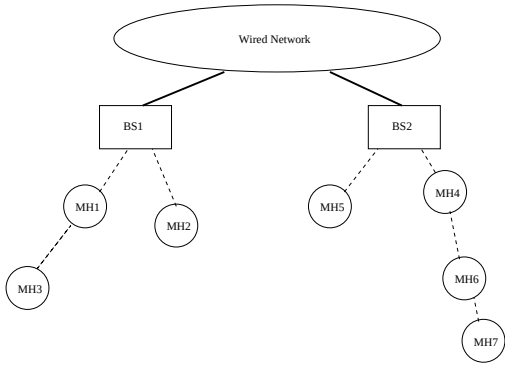

Figure 1: An example hybrid network

The protocol allows MHs to register with an AP. The "radius" around an AP is limited to a specified number of hops, K. The protocol is designed to test

the efficiency of specialized protocols when compared to more general routing protocols like Ad-hoc On-Demand Vector Routing (AODV) [1] and Dynamic Source Routing (DSR) [2]. We believe the K hop paradigm will be useful in small, busy geographic areas, such as a university campus or airport. In such areas, there is usually a limited range of coverage with respect to the entire area. For example, there may be APs inside a building, but users just outside the building cannot access it. However, if there are other mobile users, or dedicated relays, in the lobby of the building, such devices may serve as routers to extend the range of the AP to the 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. The project also lends itself to upper layer extensions, like TCP and security. Additionally, users may potentially find the protocol useful in real-world applications.

Our initial goal in designing the protocol is to have a minimal routing implementation that is complete, but not too complex. Essentially, we wanted to extract from existing research only what is required for a functioning routing protocol while not worrying about the many optimizations which may be possible. The protocol was also designed to function with standard forwarding table implementations. Standard forwarding is a function of three parameters: the destination (and netmask), the next hop and the interface. When a data packet arrives it should only be forwarded based on these criteria. Therefore, the protocol could not use any type of source routing or forwarding based on the packet's $\langle$*source, destination*$\rangle$ tuple.

The rest of this report is organized as follows. In Section 2, we discuss previous work in ad hoc routing and hybrid systems. We provide a definition for K-hop networks in Section 3. Section 4 describes how our protocol performs routing. Section 5 discusses how IP addresses are assigned in our network. In Section 6, we describe how Internet connectivity is achieved. Section 7 presents other software used in our implementation. In Section 8, we conclude the report.

2 Related Work

Our proposed routing protocol borrows characteristics of the AODV protocol [1, 3]. Consequently, we provide a brief overview of the AODV protocol in Section 2.1. DSR [2] is another widely researched reactive protocol, performing route discovery using RREQs and RREPs. DSR maintains full path information for routing as opposed to AODV which remembers only next-hop addresses for destinations. We chose to avoid maintaining source routes due to complexity issues with the Linux kernel and instead opted for following AODV's style for maintaining routes. AODV contains many optimizations and features that make it quite complex. Since one of the primary goals of our protocol is simplicity, we did not incorporate many similar features into it. Some of these features include multicast support, unidirectional link support, expanding ring search, hello messages and local repair.

Our protocol is also akin to the Zone Routing Protocol (ZRP) [4] in that it contains both proactive and reactive mechanisms. However, ZRP uses proactive routing within a zone, and reactively exchanges routing information between zones, whereas in the protocol we present, access points proactively maintain routes to reachable mobile devices, with local route discovery being completed in a reactive fashion.

Yet another protocol that achieves similar same goals as ours is LUNAR [5]. The Lightweight Underlay Network Ad-Hoc Routing (LUNAR) protocol implements a layer in between the MAC layer and the IP layer to perform a variation of multi-hop ARP. The protocol is simple and is designed to work in small-hop environments. LUNAR uses a combination of reactivity and proactivity for route discovery and maintenance. The protocol performs automatic address assignment and supports Internet access, a similar objective of our protocol. A special *gateweying node* handles passing packets back and forth between the Internet and the LUNAR network. Our protocol differs from LUNAR in the following ways:

- We do not support multi-hop ARP, and use DHCP for address assignment rather than a randomized scheme.
- The LUNAR model requires encapsulation of data packets, which we chose to avoid due to the amount of overhead incurred by passing packets back and forth between user space and the kernel.
- In LUNAR, broadcasts are performed using a tree; we use a traditional broadcast scheme.
- We employ AP broadcasting to identify nodes in each zone, LUNAR does not have any registration process.

The approach we take of combining small, K-hop ad hoc networks with a structured backbone has been

studied by several others [6, 7, 8, 9, 10, 11]. The focus of [6] and [7] is on relieving congestion in cellular networks by introducing ad hoc functionality among mobile devices. We are not explicitly concerned with network congestion. In the Multihop Cellular Network (MCN) [8] architecture, nodes in a cell are allowed to use ad hoc mechanisms to improve throughput or reduce the number of needed base stations. A-GSM [9] allows devices to relay through each other to reach a base station, in an effort to reduce the occurence of dead spots in the network. A performance comparison between ad hoc and cellular networks is presented in [10], along with a suggested hybrid system that switches back and forth between cellular and ad hoc modes based on network conditions. Our model requires all nodes to operate in the ad hoc mode, including the APs. Finally, in [11], the authors offer enhancements for ad hoc networks to increase overall throughput. One of the additions, called *assisted scheduling*, aims at using a central base station to coordinate flows between ad hoc nodes. Our model relies on the DCF mode of IEEE 802.11 [12] for packet communication and does not impose any strict scheduling.

2.1 AODV Overview

The Ad-Hoc On-Demand Distance Vector (AODV) [1, 3] routing protocol uses a packet exchange to establish routes. A source node S wishing to communicate with a destination node D broadcasts a *Route Request* (RREQ) packet and expects to receive a *Route Reply* (RREP) either from D or from an intermediate node along a path from S to D. The RREP is unicast back to the source. Upon receiving the RREP, S can begin sending data to D using the path that has been set up during the route discovery process.

Routes generated using the route discovery process are temporary and expire after some time if not recently used. If S's route to D is deleted and S has additional data to send to D, the route discovery process is initiated again. Nodes forwarding RREQs and RREPs use the route information contained in the packets to learn routes to other nodes. These routes are stored temporarily in a cache and often have a shorter lifetime than the routes stored at the source and destination nodes. To avoid processing old packets, each broadcast packet is uniquely identified by a $\langle$*source, packet_id*$\rangle$ tuple. Sequence numbers are also used to determine the freshness of routes.

AODV uses *Route Error* (RERR) packets to signal nodes of unreachable destinations. When an active link breaks, the node upstream of the broken link informs its upstream neighbors by issuing a RERR. The RERR is propagated by each upstream node depending on its own cached routes. Nodes that receive a RERR may elect to initiate route discovery for the node that is determined unreachable using the invalid route.

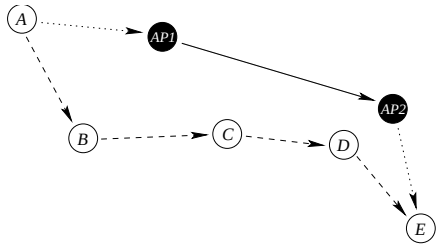

Figure 2: A sample K-hop network with $K = 4$. Node A can reach node E using either APs $AP1$ and $AP2$ or nodes B, C, and D.

3 A K-hop Network

In this section, we describe in detail the properties of a *K-hop network*.

A K-hop network is an ad hoc network where established connections between two hosts are limited to K wireless hops. Connections between two nodes that are farther than K hops away are not permitted (except for example in emergency situations where K could be ∞). To support paths that are longer than K hops, nodes must make use of APs to "jump" from one part of the network to another. Using APs, paths containing up to $2K$ wireless hops are possible. Take for example the network of Figure 2, with K equal to 4. Shown in the figure are two valid paths from node A to node E. The path indicated by the dotted and solid lines represent the path constructed by using the two APs that are closest to A and E. The total number of wireless hops in this path is 2. The second path that goes through nodes B, C, and D comprises 4 hops. Note that if K were 3 in this Figure, A would no longer be able to reach E using the path $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$, and would be limited to the first path $A \rightarrow AP1 \rightarrow AP2 \rightarrow E$. The choice of which path to use is determined by the routing protocol, such as the one presented in Section 4. As another example, for $K \geq 3$ in Figure 3, the path $A \rightarrow B \rightarrow C \rightarrow AP1 \rightarrow AP2 \rightarrow D \rightarrow E$ is valid even though the total path length is 5 since $AP1$ is reached after only 3 ($\leq K$) hops.

The *zone* for each AP is likewise defined by the nodes that can be reached in K hops or less. Looking

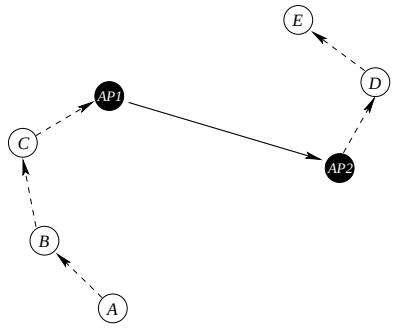

Figure 3: A valid path from A to E for $K \geq 3$.

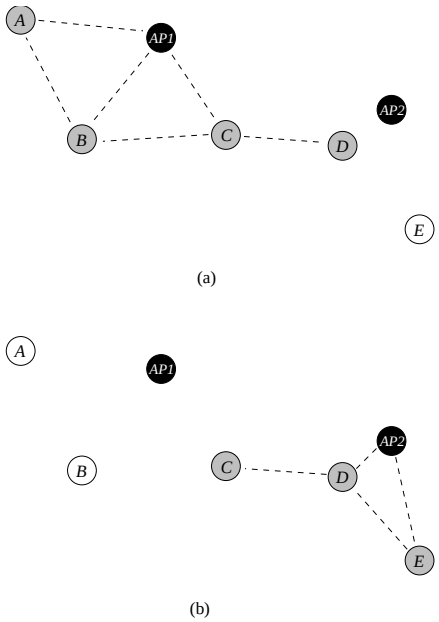

Figure 4: Zones for (a) $AP1$ and (b) $AP2$ with $K = 2$. Dashed lines indicate connectivity in the zones.

at the 2-hop networks of Figure 4, we can see the number of nodes serviced by an AP varies depending on node locations and behaviors (such as grouping). The shaded nodes in Figure 4(a) are within the zone of $AP1$, while the shaded nodes in Figure 4(b) can be serviced by $AP2$. Note that nodes C and D could use either $AP1$ or $AP2$ (or both), if all nodes and APs use the same channel.

The APs that are a part of the K-hop network are assumed to be connected via some other network medium such as highspeed wireline or wireless antennas operating in a different frequency range than the subscriber nodes. Thus, APs are able to communicate with each other to exchange information as detailed later in Section 4.

It should be noted that the value of K can be dynamic, changing over the life of the network. A large network can also potentially be made up of several small K-hop networks, each with its own K bound. We intend to determine in the future the role that K plays in network performance.

The architecture here of a K-hop network will be assumed for the remainder of the report, unless otherwise noted.

4 The Routing Protocol

Our routing protocol attempts to take advantage of a combination of reactive and proactive components similar to ZRP. The AP maintains data about routes to all MHs within its own zone.

Each AP tries to proactively discover devices in its own zone and then uses this information in answering RREQs. The collection of MHs are the reactive component, only maintaining routes to devices which they are actively using, either as a source or intermediate hop. The routing protocol essentially consists of four separate phases:

1. **Beaconing**: The process by which APs learn of MHs within their zone.
2. **Route Discovery**: Occurs when a device needs to send data to a destination for which it does not have a route.
3. **Error Handling**: The process by which the protocol handles route errors.
4. **Data Forwarding**: This phase is concerned with forwarding data packets.

All of these phases can operate concurrently, and are described in more detail in the following subsections.

The routing protocol is slightly different for APs and MHs. In this section, we describe how each phase is handled and differences in how APs and MHs respond. It should be noted that the routing protocol is designed to function in this hybrid network as well as in a purely ad hoc setting. The MHs essentially run a stripped down version of AODV. The APs respond differently by taking advantage of proactive state and connectivity to other APs whenever possible.

In our implementation, APs are actually laptops that have both a wireless and wired interface. The WLAN cards can either be in ad hoc or managed mode. Managed mode requires a MH to connect to an existing AP within one-hop. When cards are in one mode, they do not communicate with cards in

the other mode. Since our system requires cards to communicate with the AP as well as ad hoc nodes, we placed all cards in ad hoc mode and used the wired interface of the laptops, functioning as APs, to connect to the rest of the network.

4.1 Beaconing

To make APs in a K-hop network aware of the MHs within its own zone and allow MHs to create default routes (described in Section 6), our routing protocol employs an AP-driven beaconing mechanism. This beaconing process is performed at the network layer and involves a three-way handshake between MHs and APs.

The beaconing process works as follows. Each AP is responsible for periodically transmitting a *Beacon* (BEACON) packet. These beacons are broadcast with a Time-To-Live (TTL) of K. When a MH receives a beacon, it responds with a *Join* (JOIN) packet under two conditions. First, if the MH is not registered with any other APs and is not waiting for a reply from an AP due to a recently sent JOIN, it attempts to join the zone of the message originator by issuing a JOIN message. The MH will also attempt to join a zone if a BEACON originates from the AP with which it is currently registered. This soft-state leasing mechanism does not require MHs to unregister with an AP when it leaves the zone.

When the AP receives a JOIN request, it adds or updates its internal state for the MH. It then responds with a unicast *Join Acknowledgment* (JOIN-ACK) packet. Once the MH receives this message, the registration is complete.

APs drop any overheard BEACONs or JOIN-ACKs (e.g., due to overlapping coverage). Intermediate MHs that receive JOINs or JOIN-ACKs update their own forwarding tables as described in Section 4.3.

4.2 Beaconing Alternatives

Depending on the characteristics of the network, alternative beaconing schemes can be employed. The first beaconing alternative to the one described above is initiated periodically by each MH. The second method we describe is a beaconing system that is on-demand (i.e., MH data driven). Although we describe these two alternatives, we have presently only implemented the scheme described in Section 4.1.

4.2.1 Mobile Host Periodic Registration

Periodic beacons from the AP is an expensive operation, particularly when nodes are relatively stationary. Rather than making the AP responsible for identifying who is in the zone, we can make it a chore of each MH to reveal itself. In detail, when a MH first enters the network (either by booting up, or moving), it broadcasts a JOIN (with TTL = K) and receives a JOIN-ACK from one or more APs, from which a default route can be constructed. Once the IP address of the AP has been identified, subsequent periodic JOINs sent by the MH can be unicast rather than broadcast. In the event that a JOIN does not elicit a JOIN-ACK from the AP, several attempts can be made before giving up. This registration scheme has the added benefit that other nodes in the network can learn about the presence of a new node when it receives a new JOIN.

4.2.2 On-Demand Registration

The third option is to support only implicit registration with the AP when a MH wishes to send data. When a MH has data to send, it performs a RREQ. Upon receiving a RREQ from a node that is has not heard from before, the AP sends a message telling the MH to register. Through this exchange the MH learns its default route. This scheme scales to the amount of data connections in the network. This scheme suffers one major drawback. Suppose node A in zone Z_1 of AP_1 wants to find node B in zone Z_2 of AP_2. A RREQ must be flooded to *all* zones if none of the APs know about B. In contrast with the two proactive registration schemes, multiple zones incur extra traffic from the RREQ generated by A. Requiring each node to register with the AP regardless of whether or not they have data to send avoids this problem.

4.3 Route Discovery

Route discovery is currently only initiated by MHs as APs in our model do not produce application layer data. The process begins when a user space data packet is received in the kernel and the forwarding table has no entry for the destination. This data packet, and subsequent data packets for the destination, are then buffered as described in Section 7.1.

In response to this event, a RREQ packet is generated with a TTL of K. This RREQ is broadcast by the MH. The RREQ specifies the destination being searched for and the originator's IP address. The packet also specifies the current sequence number for the source which is used to avoid loops. The RREQ also includes how long intermediate nodes should cache a route to the original source. If the originating source had an entry for the requested destination

within a specified amount of time, it will remember the destination's sequence number and include it in the RREQ. If the destination sequence number is present, this indicates to intermediate nodes that they may respond on behalf of the destination if they have a fresh enough route. Otherwise, the RREQ can only be answered by the specified destination. This broadcast should not be expensive for small values of K because it can only propagate K hops. Due to the limited propagation of broadcasts, responding from cached entries is not as important. Furthermore, this avoids some loops that may occur due to routes expiring at different times as discussed in Section 4.3.1.

When a MH receives a RREQ, it first updates or adds an entry for the original source in case it must forward a RREP back. If the MH is the ultimate destination, it will generate a RREP with its current sequence number and return it to the originator. If the MH is not the destination, it will check to see if it can satisfy the RREQ. For an intermediate MH to generate a RREP, three conditions must be satisfied:

1. The MH has a route to the queried destination.
2. The distance between the MH and the originating source plus the distance from the MH to the queried destination does not exceed K hops.
3. The RREQ specifies a sequence number for the queried destination and the MH's entry is fresh enough to respond.

When RREPs are received, MHs update their entries to the destination in the same way that source entries are updated for RREQs. These entries are temporary, they are purged after a specified timeout unless data is forwarded to the destination.

APs handle the route discovery process differently because they have proactive knowledge of all MHs within their zone. For this, we use the beaconing scheme described in Section 4.1. Furthermore, we assume each AP has a well-known list of all other APs on its subnet, so it can unicast packets to them. Alternatively, if all the APs were on the same LAN, most of the control packets could just be broadcast.

When the AP receives a RREQ, it first determines if the request came from a MH or another AP according to whether the IP source is one of the well-known APs or not. Alternatively, this could be done by looking at which interface the packet came in on. The AP will only handle RREQs from MHs which are in its zone. The set of MHs within an AP's zone is determined by destinations which have responded to the beaconing recently. So for a given RREQ, multiple APs will not try to resolve it. First, the AP determines if it knows of a route to the destination. If the AP does not know of a route it will drop the packet if it is from another AP or send the RREQ to all the other APs if it is from a MH. It does not rebroadcast the RREQ over its wireless interface in this case. If the AP responds to an RREQ with a broken route, the normal RERR procedure will take place when the source tries to send data. This scenario is similar to an intermediate node responding with an invalid route to a destination.

Next, consider the case where an AP does have an entry to the destination and it can satisfy the RREQ (that is the originator allows cached responses and the sequence number is fresh enough). In this case, the request is just unicast back to the source, which may be on the wireless or wired interface.

When a MH receives a RREP for an outstanding RREQ, the forwarding table is updated immediately, and all packets are sent to the destination.

4.3.1 Sequence Numbers and Loops

Sequence numbers are an important mechanism, proposed in AODV, for providing loop-free routes. Their main function is to act as a virtual timestamp for the system to give a causal ordering to the sequence in which routes to a given destination are discovered. A useful property of sequence numbers is that as you move from the destination to a source, the sequence numbers for the destination along the path are non-increasing. However, this scheme requires state to be maintained for other MHs. This is soft-state and may timeout eventually, which means that a MH will "forget" how fresh the last route it had for a given destination was. The state is also lost when a MH crashes and reboots. This is why our protocol requires RREQs to be answered by the destination when the originator has no previous state for the destination. If a RREP is generated by the destination, a loop cannot exist. However, if every RREQ has to be answered by the destination, efficiency is lost by not allowing intermediate nodes to reply on behalf of the destination. Examples of how loops can occur in AODV are illustrated in [13].

4.4 Route Errors

A route error will occur for two reasons. The first is when a data packet is received from another MH with an IP destination that is not in the local forwarding table. Such a situation may occur when one MH's forwarding entry times out before another's. The second situation is when an MH detects the loss of communication with a neighbor in its forwarding table. A MH considers a link to be down whenever

the MAC layer has to drop a packet after multiple retransmissions. In both cases, the error must be propagated toward the source of the data packet.

Within our implementation, the information we get when one of the two errors occurs is not as complete as we would like. In the case where a destination does not exist in the table, we receive the original IP source and ultimate IP destination of the data packet. In the case where link layer loss is detected, all we are given is the MAC address of the next hop for the data packet.

Another way to detect link loss is to require *hello* or *heartbeat* messages to be periodically broadcast with a TTL of 1. This indicates that communication with a neighbor is possible. This approach is rather heavyweight as it requires another control message and a periodic broadcast. Also, the latency of learning about the downed link is usually greater.

When a data packet arrives and there is no forwarding table entry, the MH will generate a RERR, unless default routes are used (described in Section 6). The RERR simply specifies the unreachable destination IP address. If the MH has a route to the IP source, the RERR is unicast, otherwise it is broadcast with a TTL of K hops.

When a data packet is dropped at the MAC layer, a RERR is created with the unreachable neighbor's IP address and sequence number in it. Additionally, all destinations for which this neighbor serves as the next hop are also removed from the forwarding table and placed in the RERR packet. This packet must be broadcast because we have no knowledge of who the original sender was. The broadcast is done with a TTL of K. This procedure could be optimized if the WLAN card driver was modified to specify the source of the dropped packet.

When a MH receives a RERR, it will update its forwarding table. This is done by removing entries which use the RERR sender as the next hop to the specified destinations. This MH then creates a new RERR packet based on its unreachable destinations. If there is at least one unreachable destination, the new RERR packet is broadcast with the TTL decremented by one. If the TTL becomes zero, the packet is dropped. In this manner, the RERR will eventually reach the original data sender (assuming a RERR is not lost along the way and the original sender is reachable).

The only difference in the way APs handle errors is when the AP generates and broadcasts RERRs, they are sent on the wireless link as well as to each of the other APs. When APs receive RERRs from other APs, they will broadcast or unicast the message within their zone after the appropriate updates. Any broadcast after a RERR is received from an AP is done with a TTL of K. When APs receive RERRs from MHs in their zone that are broadcast, they will broadcast the RERR on the wireless interface and send it to all the APs.

There are a couple of alternative methods to propagate RERRs toward a source. First, as in AODV, a predecessor list for each destination could be maintained. This would specify the subset of neighbors that uses a MH to communicate with each destination. This subset is all the neighbors for which a MH forwards or generates a RREP. With this information, we could unicast RERRs only to the neighbors which are known to use the link. They can propagate the RERR further back in a similar manner. Another alternative would be to maintain a list of sources that use a MH for a given destination. For each source, the MH maintains a special reverse pointer to the source. This pointer is not placed in the forwarding table or used in routing, but is only used to provide a path from the MH to the source in the event of a RERR. In this case, the RERR can always be unicast to all the sources that use a given destination and all intermediate nodes along the path could also delete the unreachable destinations from their forwarding tables. This scheme would require the maintenance of these special reverse pointers.

4.5 Data Forwarding

Data forwarding should be done independent of our protocol, which is responsible for setting up the forwarding tables. Once this is done, the forwarding is done in the kernel with no intervention from our system. As discussed in previous sections, the only time data forwarding interacts with our system is when data packets arrive for a destination that is not in the table or the MAC layer times out because of too many retransmissions. Data packets for unknown destinations are detected by the method described in Section 7.1. In Section 7.2, we discuss how MAC layer time outs are detected. Since the forwarding table specifies an interface, there is no special handling for APs. Each forwarding entry will expire after a certain amount of time. If an entry is about to expire but it has recently been used to forward data, it is refreshed.

5 Address Assignment

For most networks, a node should be allowed to enter and leave at will. Therefore, MHs should be capable of being dynamically configured by the network upon

entry. Automatic address assignment in IPv4 networks is often handled by the Dynamic Host Configuration Protocol (DHCP) [14]. DHCP 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, and can thus 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.

One proposal for generating automatic IP addresses in ad hoc networks is presented in [15]. The technique presented requires little or no user configuration, a primary goal of the *Zeroconf* working group of the Internet Engineering Task Force. To obtain an address, a node first selects two random address A_1 and A_2 from the 169.254/16 address range. A_1 is used as a temporary address for the subsequent address request procedure, and A_2 indicates the candidate address the node wishes to obtain. A new ICMP message called an AREQ is broadcast to all the nodes in the network by the node requesting address A_2. If someone else in the network is already using A_2, a AREP is generated and unicast back using the temporary address A_1. If no response is heard, the requesting node assumes that the address A_2 is not in use and is therefore valid. This method for performing *Duplicate Address Detection* (DAD) splits the available address space in half, severely reducing the number of hosts that can be a part of the ad hoc network. A second problem arises if two nodes pick the same temporary addresses in separate network partitions that later merge [16].

Another attempt at providing addresses for ad hoc nodes is presented in [17]. The paper describes a distributed dynamic host configuration protocol for configuring nodes in a MANET. Essentially, when a new node enters the network, a different node proposes a candidate IP address for it and an agreement is made among all the nodes in the ad hoc network. An assumption of this solution is that the ad hoc network is a *stand-alone* network, with no external connections, whereas our model assumes constant global connectivity. Furthermore, sending a message to every node in the network for each assigned address results in unnecessary packet overhead.

The schemes described so far are adequate for ad hoc networks that are isolated from the rest of the Internet. When Internet connectivity is available, the addresses selected using either algorithm might not be globally valid. We believe that using a protocol such as DHCP in this setting both reduces the amount of overhead as compared to the other proposals, and allows for assignment of useful IP addresses. Additional benefits of using DHCP include being able to perform load balancing, network security and monitoring, and faster and more accurate routing. Below we describe how DHCP *can* be used in ad hoc networks that are connected to a central DHCP server via APs.

5.1 Obtaining an Address

For nodes that are only one hop away from the AP, the DHCP protocol works fine. A problem arises for nodes that are more than one hop away. To allow for these nodes to obtain an IP address from the server, we require that *all* nodes in the ad hoc network run the DHCP relay daemon after obtaining their own address.

The steps used for obtaining an IP address are identical to those used in the standard DHCP protocol. 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.
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 starts the DHCP relay daemon 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 relay 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. Since the node *did not* perform route discovery for the DHCP server, a route must be added by the system to indicate that the route to DHCP server uses the same next hop as that of the node's associated AP. Assuming that nodes are equipped with default routes as described in Section 6, address discovery for the DHCP server can be suppressed in favor of the default route. On the occasion where the route to the AP fails, route discovery is performed to find a new path. The AP, upon hearing a RREQ for the IP address of the DHCP server, should respond on behalf of the DHCP server.

This form of address assignment runs under the stipulation that nodes closer to the AP must obtain valid IP addresses before nodes farther away can contact the DHCP server. Since address assignment is the first step when a node boots up or restarts, this requirement seems reasonable. The delay experienced by a node requesting an address is relatively low, except in the case where it is not within reach of the AP and the DHCP request times out, in which case we recommend using a scheme such as the one in [15] for creating temporary addresses.

5.2 Alternatives

The above address assignment technique attempts to minimize latency for obtaining an address. The scheme can be modified to meet other needs. Below we discuss some alternatives to our basic scheme that could be implemented to reduce either operational cost or network overhead.

1. If we assume that the DHCP server always resides behind the APs, it is not necessary for nodes to know the IP address of the server. Instead, all DHCP traffic can be directed toward the AP instead. The AP, upon receiving a DHCP message, forwards the message to the server. The trade-off here is more complexity at the AP in exchange for protocol simplicity and decreased latency.

2. Broadcasting DHCP DISCOVER messages while having multiple relay neighbors can result in unnecessary redundant network traffic. To avoid this situation, a node wishing to obtain an IP address can decide to wait until at least one beacon has been heard from a nearby AP. This modification requires the AP beacons to contain the IP address of the DHCP server. Once this address is learned, the node can bypass broadcasting DHCP DISCOVER messages in favor of unicasting DHCP REQUEST messages to the server.

With our K-hop architecture, it is not necessary to perform DAD when a node enters a new zone. Our model assumes that addresses are unique across all zones, since they are allocated by one central DHCP server. The one scenario where DAD might be required is when an isolated ad hoc cloud comes in contact with a zone.

6 Internet Connectivity

A goal of our protocol is for ad hoc nodes to be able to communicate with hosts on the Internet. Attaching ad hoc networks to the Internet requires solving several addressing and routing issues. Several methods for achieving Internet connectivity have been offered already [18, 19, 20, 21].

An initial design proposed by Broch *et al.* [18] attempts to integrate the DSR protocol with Internet routing and Mobile IP [22, 23]. In this technique, foreign agents act as relays between the ad hoc and wired networks. Nodes inside the ad hoc network are given legally routable addresses from a single IP subnet, so that outside routers can advertise their presence.

Similar designs employing AODV have been offered [21, 19, 20]. These designs use Mobile IP foreign agents as gateways to the Internet. In MIP-MANET [21], nodes are required to use their home address for all communication, whereas in the approach taken in [19, 20] requires nodes entering an ad hoc network to obtain a care-of-address either from a foreign agent (FA) or by automatic generation using an advertised network prefix. In the second case, the address must be checked for uniqueness using a form of DAD such as the one proposed in [15].

In the schemes described thus far, a node that receives a packet for which it has no next-hop entry will generate a RERR. Therefore, when an internal ad hoc node, S, wishes to connect to a host, D, on the Internet, every node along the path from the source to the FA must also have a route to D. This requirement can result in crowded routing tables. Although each ad hoc node bears the burden of being a router for other nodes in the network, they should not have to keep track of hosts that are outside of the ad hoc network.

To circumvent this situation, once the route discovery reveals that the path to D is through the FA, S can use IP encapsulation to forward data packets first to the FA, which then decapsulates the packet and relays the data to D. On the reverse trip, data for S is re-encapsulated and forwarded using the ad hoc protocol to S. This approach is adopted by both [21] and [20].

An alternative for dealing with Internet host routes is to introduce the concept of *default routes* for each ad hoc node, as suggested for IPv6 ad hoc networks [24]. In this technique, each ad hoc node maintains a special route toward its *Internet-gateway*, a router placed at the edge of the ad hoc network that provides Internet connectivity. This default route is used when a node receives a data packet for which

it does not have a host-specific next hop. As such, each node along the path from S to D does not need to know about D, and IP encapsulation is no longer necessary. However, this scheme has drawbacks for dealing with link breakages and repairs efficiently.

Our method for providing Internet connectivity to nodes in the ad hoc network is based on [24], with several changes for operation under IPv4 and without the need for mobility agents. Below we describe suggested protocol changes in order to achieve Internet access.

6.1 Routing Protocol Modifications

To support routing to hosts on the Internet, we have modified our routing protocol in Section 4. In this section, we describe the differences between the basic and modified protocols.

6.1.1 Modified Route Discovery

The route discovery process for destinations that are within the ad hoc network (determined by the subnet mask of the destination) is identical to that of Section 4.3. This process may be repeated several times by S before S concludes that the node is unreachable.

If the subnet mask for D does not match that of the ad hoc network, S attempts to use its default route (if one is available) to reach D. Recall from Section 4.1 that a node's default route is synonymous to the path used to reach its AP, and is discovered during the beaconing process. If no such route exists, S temporarily concludes that D cannot be reached. On the other hand, if S does have a default route, data packets are forwarded to D using this path.

Processing of RREQs by APs is unchanged from that in Section 4.3. If an AP has no knowledge of the destination specified in the RREQ, it queries the other APs in the network, one of which will respond if the destination is valid. When replying to a RREQ, APs are recommended, but not required, to send gratuitous RREPs to the destination to avoid unnecessary route discovery by the destination for the source in the case of protocols that require feedback from the destination (e.g., TCP).

6.2 Modified Data Forwarding

For incoming external packets (i.e., packets that originate from a host outside of the ad hoc network), an AP A checks its routing table to see if it has a route to the destination. If a route does not exist, A performs route discovery for the destination. An ICMP [25] unreachable message is generated and sent to the host on the Internet if A does not receive a RREP after several attempts. If a RREP is heard, A will forward the packet using the newly discovered route. This operation is only necessary if each ad hoc node has a global address. Assignment of global IP addresses for ad hoc nodes in IPv4 can be achieved by enabling DHCP [14] to operate over multiple hops using the method of Section 5. If the ad hoc nodes are given private IP addresses, A should not expect to receive any incoming packets for which it does not have an associated connection already established.

Outgoing external packets (i.e., packets destined for hosts outside the ad hoc network) in our protocol are forwarded using the default route. Intermediate nodes, upon receiving a packet with an unknown destination, simply use their own default route to forward the data. If a node cannot forward the packet using its default route (e.g., due to link breakage), a RERR is generated, indicating that the default route is outdated, and must be recalculated. Packets that reach the AP are simply forwarded on to A's default gateway. From this point on, an ICMP destination unreachable message is issued to the sender in the event of an error.

7 System Software

The implementation of our protocol used open-source software from two other sources to address our needs. The Ad hoc Support Library (ASL) [26] provides a general API for use in systems such as ours. Its major functions are to efficiently handle data packets which do not have an entry in the forwarding table and to determine how recently a forwarding table entry has been used. The Wireless Tools for Linux [27] gives a common API for interacting with wireless devices. It is supported by widely used WLAN cards including Cisco's Aironet and Orinoco's WaveLAN cards. This allows us to detect link loss and extract other link characteristics from the driver without worrying about the specific underlying hardware.

7.1 Ad hoc Support Library (ASL) [26]

The ASL provides a simple interface to handle data packets which trigger events in ad hoc protocols. First, a default route entry is placed in the forwarding table which goes to a virtual tunneling device (TUN/TAP). Therefore, any packet which uses the default route is passed to this device, transferring it from kernel space to user space. This sets a polled file descriptor. If the IP source is from the local machine, our protocol initiates route discovery. If the

IP source is from another destination, a route error is generated. While in the process of route discovery, all other packets for the destination will also be queued in the virtual device. When the route discovery is finished, the packets are either dropped if the destination was not found or re-injected into the kernel for forwarding if the entry was added. If the packets are re-injected, they are sent via a raw socket since they already have IP headers. The ASL also includes a module which records how recently forwarding entries are used and the API allows us to query this data for a given destination. By using the ASL, we do not have to pass every data packet to user space or do any kernel programming.

7.2 Wireless Tools [27]

Wireless Tools for Linux gives a common interface to the wireless device. Open-source device drivers have been developed to interact with Wireless Tools. It serves as a bridge between device drivers and user space programs. From it, we can extract data such as the address of the AP a card is currently associated with, the signal level of the AP and the current transmission power level. Our protocol uses it for two purposes. First, we use it to identify which interface is wireless when setting up the sockets. The second use is for detecting link layer loss. While the mechanism they provide for this is rather convoluted, it is much easier than trying to manipulate the device driver. The interface provides a socket we can poll for link loss and when it occurs, the link layer address of the destination is passed to user-space. From this information, we can determine the IP address from the ARP cache.

8 Conclusion

We have presented a simple routing protocol that extends the connectivity of APs while avoiding AP use when it is not necessary. Furthermore, the system limits the range of broadcasts and trades reachability with overhead. The protocol also bridges a gap between reactive and proactive components in a wireless system. We allow efficient reactivity in the ad hoc part and proactivity between APs, where bandwidth is plentiful, to reduce latency and avoid additional broadcasts.

This report describes work that has been completed on the project. Additional work is ongoing and will be presented in future papers.

References

[1] C. Perkins and E. Royer, "Ad-Hoc On-Demand Distance Vector Routing," in *Proceedings of the IEEE WMCSA*, 1999.

[2] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing* (Imielinski and Korth, eds.), vol. 353, Kluwer Academic Publishers, 1996.

[3] C. E. Perkins, E. M. Belding-Royer, and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing." IETF Internet Draft, *draft-ietf-manet-aodv-10.txt*, Work-in-progress, March 2002.

[4] Z. Haas, "A New Routing Protocol for the Reconfigurable Wireless Networks," in *Proceedings of the IEEE ICUPC*, 1997.

[5] C. Tschudin and R. Gold, "LUNAR: Lightweight Underlay Network Ad-Hoc Routing," tech. rep., University of Basel, Switzerland, January 2002.

[6] C. Qiao and H. Wu, "iCAR: An Intelligent Cellular and Ad-hoc Relay System," in *Proceedings of the IEEE IC3N*, October 2000.

[7] X. Wu, S. Chan, and B. Mukherjee, "MADF: A Novel Approach to Add an Ad-hoc Overlay on a Fixed Cellular Infrastructure," in *Proceedings of the IEEE WCNC*, September 2000.

[8] Y. Lin and Y. Hsu, "Multihop cellular: A new architecture for wireless communications," in *Proceedings of INFOCOM*, pp. 1273–1282, 2000.

[9] G. Aggelou and R. Tafazolli, "On the Relaying Capacity of Next-Generation GSM Cellular Networks," *IEEE Personal Communications*, vol. 8, pp. 40–47, February 2001.

[10] H. Hung-Yun and R. Sivakumar, "Performance Comparison of Cellular and Multi-hop Wireless Networks: A Quantitative Study," in *Proceedings of the ACM SIGMETRICS*, June 2001.

[11] R. Sivakumar and H. Hsieh, "On Using the Ad-hoc Network Model in Wireless Packet Data Networks," in *Proceedings of the ACM MOBIHOC*, June 2002.

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

[13] K. Bhargavan, D. Obradovic, and C. A. Gunter, "Formal Verification of Standards for Distance Vector Routing Protocols," tech. rep., University of Pennsylvannia, 1999.

[14] R. Droms, "Dynamic Host Configuration Protocol." IETF Request for Comments (Standard) 2131, March 1997.

[15] C. E. Perkins, J. T. Malinen, R. Wikikawa, E. M. Belding-Royer, and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks." IETF Internet Draft, *draft-ietf-manet-autoconf-01.txt*, Work-in-progress, November 2001.

[16] N. H. Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks," in *MOBIHOC*, 2002.

[17] S. Nesargi and R. Prakash, "MANETconf: Configuration of Host in a Mobile Ad Hoc Network," in *INFOCOM*, August 2002.

[18] J. Broch, D. Maltz, and D. Johnson, "Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks," in *Proceedings of the Workshop on Mobile Computing*, June 1999.

[19] E. M. Belding-Royer, Y. Sun, and C. E. Perkins, "Global Connectivity for IPv4 Mobile Ad hoc Networks." IETF Internet Draft, *draft-royer-manet-globalv4-00.txt*, Work-in-progress, November 2001.

[20] Y. Sun, E. M. Belding-Royer, and C. E. Perkins, "Internet Connectivity for Ad Hoc Mobile Networks," *International Journal of Wireless Information Networks*, vol. 9, April 2002.

[21] U. Jonsson, F. Alriksson, T. Larsson, P. Johansson, and G. Q. M. Jr., "MIPMANET — Mobile IP for MObile Ad Hoc Networks," in *MOBIHOC*, pp. 75–85, August 2000.

[22] C. E. Perkins, "IP Mobility Support for IPv4, Revised." IETF Internet Draft, *draft-ietf-mobileip-rfc2002-bis-08.txt*, Work-in-progress, September 2001.

[23] C. E. Perkins, "Mobile IP," *IEEE Communications Magazine*, vol. 3, pp. 84–99, May 1997.

[24] R. Wikikawa, J. T. Malinen, C. E. Perkins, A. Nilsson, and A. J. Tuominen, "Global Connectivity for IPv6 Mobile Ad hoc Networks." IETF Internet Draft, *draft-wakikawa-manet-globalv6-00.txt*, Work-in-progress, November 2001.

[25] J. Postel, "Internet Control Message Protocol." IETF Request for Comments (Standard), September 1981.

[26] V. Kawadia, Y. Zhang, and B. Gupta, "System Services for Implementing Ad–Hoc Routing Protocols," in *Proceedings of the International Workshop for Ad Hoc Networking*, August 2002.

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