Supporting Legacy Devices in Multi-hop Ad-hoc Wireless Networks

A. S. Krishnakumar, P. Krishnan, Shalini Yajnik
Avaya Labs, 233 Mt. Airy Rd., Basking Ridge, NJ 07920
{ask,pk,shalini}@avaya.com

Abstract—Multi-hop ad-hoc wireless networks need additional software at each node to provide routing functionality. There is significant benefit in supporting legacy devices (i.e., devices that do not have any additional routing software) in such networks. In this paper, we present a system and techniques for legacy device support in multi-hop ad-hoc wireless networks. One conceptual approach is to enable access point-like support at each multi-hop ad-hoc node that allows the legacy devices to associate in infrastructure mode. In contrast, we present a novel scalable architecture that extends the benefits of path redundancy inherent in multi-hop ad-hoc networks to legacy devices and also supports mobility. Using the legacy device as a message reflector in a transparent manner allows the system to make decisions in a light-weight distributed fashion. We describe our architecture, protocols and a prototype system. We also present some analysis and observations from the prototype.

Keywords: Multi-hop ad-hoc, Legacy support, Implementation, Architecture, Protocol, 802.11.

I. INTRODUCTION

Multi-hop ad-hoc wireless networks [1], [2] are expected to have a significant impact on the design and deployment of many military and civilian applications, for example, combat field surveillance, disaster management, data gathering, sensor networks, emergency applications, and monitoring. They can also form a basis for small/medium enterprise networks. Ad-hoc networks have useful properties, including ease of installation and auto-reconfiguration. Significant prior work in this area has studied various aspects of multi-hop ad-hoc networking, including routing protocols [3], [4], [5], network deployment [6], mobility [7], resource constraints [8], multicasting [9] and security [10]. The assumption has been that every node in the network can have custom software that provides the routing functionality.

An important issue in multi-hop ad-hoc networks is whether out-of-the-box legacy devices (i.e., devices without any additional installed software) can be supported. Consider, for example, a multi-hop ad-hoc network deployed in a building. A visitor or an emergency worker may like to use a portable device over this network without installing or upgrading any software on the portable. The problem that we address in this paper is how the multi-hop ad-hoc network can support such portable devices.

One conceptual approach is to support such legacy devices through the 802.11 infrastructure mode. In this model, the ad-hoc nodes themselves support the access point functionality. This approach is being investigated by the IEEE 802.11s working group [2] (and also some vendors [11], [12], [13]) through modifications to the basic 802.11 Medium Access Control (MAC) protocol [14] to allow legacy device support via mesh access points.

In 802.11 infrastructure mode, terminals associate with one access point and communicate through it. In contrast, in the ad-hoc mode the nodes in the network can talk to multiple other nodes at the same time (without any “association” issues). This feature allows the creation of multiple paths between nodes in the ad-hoc network, and is useful in several scenarios, including resource-constrained networks, low-power sensor networks, and networks that require load-balancing. This has led to innovative techniques that use the in-built network redundancy for various features [15], [16]. In this paper, we present a novel architecture that we call WaterWalker, that extends such benefits of ad-hoc wireless networks to legacy devices. As shown in Figure 1, WaterWalker allows the legacy devices to “associate” with multiple ad-hoc end-points and communicate through them to the rest of the network thereby extending the inherent redundancy of a multi-hop ad-hoc network to the legacy devices.

However, achieving this objective is challenging since there is no custom (including multi-hop ad-hoc routing) software on the legacy devices. Our approach puts additional intelligence in the nodes of the multi-hop ad-hoc network that allows these nodes to monitor the arrival and presence of legacy devices. The connectivity of the legacy devices is channeled through one or more proxy nodes in the ad-hoc network. The choice of appropriate proxy nodes and the mobility of legacy devices are handled seamlessly through a light-weight protocol that allows decisions to be made without direct coordination between the ad-hoc nodes. The protocol also transparently co-opts the legacy devices within the protocol to achieve an
efficient distributed implementation. In the rest of this paper we use the term ad-hoc to refer to multi-hop ad-hoc networks, unless additional clarification is needed.

Section II discusses multi-hop ad-hoc wireless networks and legacy device support within such networks. Sections III and IV talk about the components and functions of the Water-Walker architecture. Section V analyzes protocol parameters and overheads. Section VI presents a prototype system and some quantitative observations based on our implementation. We conclude in Section VII.

II. LEGACY DEVICES IN AD-HOC NETWORKS

IEEE 802.11 IBSS mode [14], which forms the basis for ad-hoc networks, does not require access points and allows nodes to communicate with each other directly if they are in wireless range of each other. Multi-hop ad-hoc networks, as shown in Figure 1, allow nodes (e.g., nodes N1 and N6) to use other nodes as relays (e.g., node N5) when the two endpoints are not directly in wireless range. This is enabled through the use of a routing protocol. Routing protocols for mobile ad-hoc networks [3], [4], [5] are implemented through additional routing software at each node in the ad-hoc wireless network.

In this paper, we consider legacy devices to have a standard IP networking stack, for example, devices that run Linux or Windows operating systems. Such legacy devices that do not have routing software can be supported using proxy nodes in the ad-hoc network. In contrast to the 802.11s mesh architecture where mesh access points act as the proxy nodes, we propose a light-weight approach that is useful in scenarios requiring path redundancy that extends all the way to the legacy device.

Our WaterWalker1 architecture allows legacy devices to connect with the ad-hoc network in the 802.11 IBSS mode. The architecture is motivated not only by the desire for path redundancy but also by the fact that, when compared to legacy devices, the nodes in the ad-hoc network have more information about the state of the network and thus can make better decisions on the routes to/from the legacy device. Upon discovering the arrival of a legacy device, the nodes in the ad hoc network need to make a decision of how many and which nodes proxy for the legacy device. This has protocol and performance implications related to how the communication between the ad-hoc nodes and the legacy device takes place. For instance, a distributed election mechanism between the nodes in the ad-hoc network to choose a proxy would lead to a complex protocol with high bandwidth overhead. Instead, we propose a novel mechanism where the nodes make autonomous decisions by intelligently co-opting the legacy device into the decision-making process. This is achieved through informational messages described later. The protocol handles device mobility, in a manner transparent to the legacy device, through distributed monitoring. We would like to emphasize that all these functions are implemented without changes to the legacy device.

Once proxy nodes are chosen, they can inject routes for the legacy device within the ad-hoc routing. Any of the ad-hoc routing protocols [3], [4], [5] can be enhanced to provide this functionality.

III. WATERWALKER: COMPONENTS

We now present the details of the protocol building blocks of the WaterWalker architecture.

A. Protocol Building Blocks

The key enabler of our architecture is a set of protocols. These protocols are built out of three main message classes: trigger, informational and command messages.

Trigger Messages, \( T \). Trigger messages are basic messages that activate the generation of other types of messages, specifically, command and informational messages described below. Parameters in the trigger message determine the type of command or informational message that is generated. In many cases, trigger messages are specifically crafted by the nodes in the ad-hoc network.

Informational Messages, \( I \). Informational messages are multicast messages sent from a legacy device that provide useful information to nodes in the ad-hoc network. One example of useful information is the current link quality to the legacy terminal. The signal strength of the received informational message can provide an indication of the link quality. Informational messages can also encode specific data.

\[ \text{Legacy} \quad \text{N1} \quad \text{N2} \quad \text{N3} \]

\text{Trigger Message (Data)}

\text{Informational Message Multicast (Data)}

Fig. 2. Reflective Announcements.

There are some key aspects of informational messages that are worth noting. Firstly, these messages are sent from the legacy device, and may be triggered using a trigger message from an ad-hoc node. We call this technique reflective announcement. Figure 2 gives a pictorial view of the technique. The reflective announcement technique is an important component of our system and is designed based on the observation that the information about the legacy device is of most interest to ad-hoc nodes in the wireless vicinity of this legacy device. While this information in some cases could have been propagated over the ad-hoc network, doing so might require several broadcasts over the ad-hoc network to ensure that all nodes in the vicinity of the legacy device receive the information. Co-opting the legacy device makes this process efficient. The challenge here is to implement trigger and informational messages without additional software on the legacy device, and the solution is outlined later in this section.

1Like legacy devices using the services of the ad-hoc network without being a part of it, Water Walkers (Water Striders) are insects that walk on water while causing only minimal local disturbance.
Command Messages, C. Command messages are always directed messages sent from an ad-hoc node to a legacy terminal. Command messages could be used for several purposes. In our system, they are used for route population. Command messages are generated in several situations: either in response to a trigger or informational message from a legacy device, or initiated by the ad-hoc node due to perceived changes in the network and environment. The specific way in which command messages are generated and used will be clear from the discussion in Section IV.

B. Implementation of Protocol Building Blocks

The protocol building blocks can be implemented using standard (unmodified) IP stacks in legacy devices, by carefully choosing and manipulating parameters for Address Resolution Protocol (ARP) [17]. ARP is used in standard networking stacks to translate a Layer-3 address to a Layer-2 address. An important point used later is that both an ARP request and a response can be used to populate the ARP table at a host and are suitable command messages. One possible implementation of reflective announcements is through ARP. In the IP context, an ARP request acting as a trigger message, with a pre-set IP i and multicast MAC m used as the sender address will generate an ARP reply from the legacy device targeted towards MAC address m. The value of m and i can also encode the data in the informational message.

IV. WATERWALKER: FUNCTIONS

The main WaterWalker functions are elaborated in this section.

A. Discovering a Legacy Device

When a new legacy device appears in the wireless vicinity of an ad-hoc node, the ad-hoc node detects its presence through broadcast/multicast messages initiated by the legacy device. These messages could be Dynamic Host Control Protocol (DHCP) messages, ARP messages or any other type of broadcast messages. Once an ad-hoc node detects the presence of a new legacy device, it initiates the protocol to determine if it needs to proxy for this legacy device.

B. Choosing the Proxy

Broadcast messages discussed above in Section IV-A trigger the protocol for choosing the proxy at the ad-hoc nodes. Choosing the proxy involves two steps: (a) determining if the ad-hoc node is a good proxy node, and (b) communicating the proxy information to the legacy device, which involves installing (host-based) route(s) at the legacy device. For example, a received DHCP message activates step (a), while a received ARP request activates both steps (a) and (b).

In step (a), the ad-hoc node can use a combination of several metrics, including signal strength of the received signal from the legacy device and current load on the node, to determine whether it is capable of proxying for the legacy device. Once the decision to proxy for a legacy device has been made, the ad-hoc node installs the routing entry for the device in its local routing table and advertises its ability to reach the legacy device in subsequent (ad-hoc) routing announcements. We refer to the set of ad-hoc nodes capable of proxying for the legacy device as the candidate proxy set.

Step (b) involves communicating the proxy information to the legacy device. When the legacy device wants to communicate with another node within or outside the network, it sends a RouteRequest trigger message (e.g., an ARP request) which contains the IP address of the destination node. All ad-hoc nodes in the candidate proxy set for this legacy device receive this message. In a simple protocol, all of these ad-hoc nodes would respond back with a command message (e.g., an ARP response) that installs a route at the legacy device.

Our protocol allows multiple commands for a trigger. The protocol stack at the legacy device will automatically choose one of the received messages (typically, the first or the last, depending on the message) and install a route. We refer to the set of ad-hoc nodes that have sent command messages to the legacy terminal as the installed proxy set. Not all installed proxies may have their advertised routes chosen by the legacy device. Furthermore, the set of installed proxies may be a subset of the candidate proxy set. For the simple protocol described above, the installed proxy set is the same as the candidate proxy set.

However, completely uncoordinated command messages from the proxies lead to inefficiencies, both in terms of wireless channel utilization and possibility of non-optimal routes being installed in the legacy device. To tackle this issue, we use an adaptive backoff strategy. In adaptive backoff, instead of responding with a command message immediately on receiving a RouteRequest trigger, each proxy node sets a backoff timer, and decides to respond after timer expiration. The timer can be set based on a global pre-set function that can depend on both the local perceived quality to the legacy device (e.g., signal strength) and the route metric to the destination. However, a node will respond only if no other node has already responded to the trigger from the legacy device within its backoff interval. The effectiveness of adaptive backoff is studied in Section V-B. Receiving information that another node has already transmitted a command message can be achieved through reflective announcement. In this case, a RouteAdvertisement-T trigger message can encode the IP address corresponding to the destination node for which a route was installed on the legacy device, or it can encode both the destination and the chosen proxy for that destination. The resulting RouteAdvertisement-I informational message will multicast this information from the legacy device to all neighboring ad-hoc nodes.

C. Monitoring the Legacy Device

Since nodes in the ad-hoc network (including the legacy devices) can move around, the link quality can vary over time. It is therefore necessary to continually monitor the status of the wireless link quality. The quality of the link (e.g., signal strength) to the legacy device is an important criterion for choosing a candidate proxy.
In our architecture, the ad-hoc nodes in the candidate proxy set (not just the installed proxy set) for a legacy device have an ongoing responsibility to monitor the device. This is because the nodes in the candidate proxy set may have advertised their ability to reach the legacy device as part of the ad-hoc routing protocol. The main idea for monitoring the legacy devices is to elicit keep-alive messages from the legacy device. The realization of these keep-alive messages can be achieved through the reflective announcement mechanism. A keep-alive trigger message is sent from a candidate proxy, and the resulting informational response (multicast) message is used to update information at all candidate proxies. A technique similar to adaptive backoff is used to ensure that only one candidate proxy initiates a keep-alive message, and the protocol accommodates load balancing policies amongst these candidate proxies. Computation of the required monitoring rate is outlined in Section V-A.

D. Handling Mobility

When a candidate proxy determines that it can no longer support a legacy device, it removes the legacy device from its routing table and pushes this information to the rest of the ad-hoc network. If this candidate proxy node also acts as an installed proxy for one or more destinations for the legacy device, it needs to inform the rest of the candidate proxies of its intent to invalidate itself as a proxy for the device. This is achieved through a reflective announcement. The ad-hoc node sends a RouteInvalidation-T trigger to the legacy device which elicits a RouteInvalidation-I informational message from the legacy device. The RouteInvalidation-T trigger from the ad-hoc node can either be on a per-destination basis or could invalidate all destinations. The RouteInvalidation-I informational message sent from the legacy device also acts as a RouteRequest trigger message for choosing the proxy and activates step (b) of the procedure from Section IV-B.

The implementation of installing the proxy on a legacy device in this context differs slightly from the method described in Section IV-B. For example, in the IP context, an unsolicited ARP reply is usually dropped by an endpoint. Hence, an ARP Reply message cannot be used as a route command message. The command message in this case can, however, be an ARP Request message with the sender IP address being the invalidated destination’s IP address, in the case where the invalidation is on a per-destination basis. If the invalidation message invalidates all routes, command messages for each destination need to be sent to the legacy device.

E. Achieving Path Redundancy

An important consideration that motivated the WaterWalker architecture was the ability to extend the benefits of path redundancy to the legacy device. Path redundancy is achieved by installing different proxies at the legacy device and may be used for many purposes, for example, reliability, tackling resource utilization constraints, and load balancing.

Routing of packets to the legacy device from the ad-hoc network is in the control of the ad-hoc network, and is therefore amenable to any desired protocol or policy of interest. Packets from the legacy device can be routed through different proxies based on chosen policy criteria. This is achieved using a technique similar to the one described in Section IV-D by over-writing the proxy installed at the legacy.

V. Protocol Parameters

We now compute the protocol overheads for each function discussed in Section IV. Discovery is done based on existing broadcast messages from the legacy device and incurs no additional overhead. Installing a proxy involves two steps. Determining if an ad-hoc node is suitable as a proxy (Step (a)) is done by each ad-hoc node independently based on local information and involves no additional network overhead. Installing a proxy (Step (b)) incurs 4 messages per destination for each proxy in the installed proxy set for that destination. This includes the RouteRequest trigger, the command response, the RouteAdvertisement-T trigger and the RouteAdvertisement-I informational messages. The adaptive backoff technique is designed to reduce the expected number of ad-hoc nodes sending messages, and we derive this quantity in Section V-B.

Monitoring incurs an overhead of $2k$ messages per monitoring period, $\tau$, where $k$ is the number of monitoring messages needed. In some cases, $k = 1$. In other situations, for example, to measure signal strength from the legacy device, $k > 1$ is preferable. The monitoring period, $\tau$, is discussed in more detail later in this section.

As described in Section IV-D, invalidation by an installed proxy can be either per destination, or one message that indicates that all destinations have been invalidated. On a per destination basis, an invalidation incurs an overhead of 6 messages per destination (2 messages to invalidate and 4 messages to install a destination). If all destinations are invalidated concurrently, the total overhead is $2 + 4d$ messages (i.e., 2 messages to invalidate, and $4d$ messages to install routes to destinations), where $d$ is the total number of destinations in the ad-hoc network.

To get a quantitative feel for desired monitoring periods and bandwidth overheads, we adopt a semi-engineering approach to compute parameters like $\tau$, and understand the effect of adaptive backoffs. This analysis is also helpful in choosing appropriate parameters for an implementation (including our prototype described in Section VI).

A. The Monitoring Interval $\tau$

The monitoring interval $\tau$ is related to our desired support for mobility. Monitoring provides information to an ad-hoc node (proxy) about the link quality to a legacy device. To estimate $\tau$, we consider a simplified scenario where the ad-hoc nodes are stationary and the legacy device is mobile with a maximum speed of $v$. (Note that the protocol itself can handle both legacy and ad-hoc node mobility.)

The mean signal strength $S$ at a distance $d$ from a radio source can be estimated [18] by:

$$S = S_0 - K \log d, \quad K > 0,$$

(1)
where $S_0$ is the signal strength (measured on a logarithmic scale, typically dBM) at a reference distance of 1 meter and $K$ is proportional to the propagation constant for the environment. Let $s_l$ denote the minimum mean signal strength for acceptable communication. As shown in Figure 3, in time $\tau$, the legacy device travels a distance of $v \cdot \tau$, which implies that the invalidation procedure is initiated when the signal strength falls below $s_u$.

As an example, for an indoor environment, if we assume that $K = 4$, the wireless card transmits at $100mW$, $s_l = -84\text{dBm}$, $s_u = -77\text{dBm}$, and $v = 3$ miles per hour (matching a typical walking speed), then $\tau \approx 10$ seconds.

B. Adaptive Backoff Interval

Recall that adaptive backoff is used to reduce the probability that multiple ad-hoc nodes respond to a legacy device. To derive this collision probability, assume that a trigger requires a response within $t$ seconds. Since it is desirable to complete step (b) of installing a proxy as one transaction, let us assume that the time taken to complete that step is our unit of time and denote it by $u$. Clearly, we have $t/u$ units of backoff values (or slots) that an ad-hoc node can assign for itself to wait. In a simple solution, a slot can be chosen at random. Our approach is to aggregate the slots into bins based on the expected values of the metric used. An ad-hoc node chooses a bin based on the value of its metric and then randomly chooses a slot within the bin.

As a coarse approximation, we can bound the collision probability from above by the probability $\hat{p}_c$ that there is a collision in any slot within a bin. However, an interesting aspect in computing the probability of slot collision for our backoff timer problem is that the only collision of interest is for the lowest ranked bin. We call this effective collisions. In particular, if an ad-hoc node chooses bin $x$, the collisions in all other bins with value greater than $x$ are irrelevant, since only the ad-hoc node(s) that choose(s) the lowest backoff timer will transmit.

To analyze effective collision probability, $p_c$, we assume a model with $n$ slots, $1, \ldots, n$, and $k > 1$ nodes choosing one of the $n$ slots uniformly at random. If $p_i$ is the probability that the lowest-numbered occupied slot was chosen by exactly one node, then $p_c = 1 - p_i$. To compute $p_i$, if the lowest-numbered occupied slot is $i$, then there are $n - i$ available slots for the other $k - 1$ nodes to choose. Since the node that chooses slot $i$ can itself be chosen in $k$ ways, and $k$ nodes can randomly choose from $n$ slots in $n^k$ ways, $p_i = 1 - (k/n^k) \sum_{i=1}^{n-1} i^{k-1}$. In contrast, the coarse analysis yields $\hat{p}_c = 1 - n!/(n-k)!n^k$.

In practice, the value of $t$ (e.g., ARP request timeouts) is of the order of several seconds. Our prototype implementation shows that $u$ is a few milliseconds. Hence, the number of slots, $t/u$ is approximately several thousands. If we group the slots into a few (e.g., 5) bins based on ranges for the metric, each bin will have several hundred slots. With 100 slots/bin, and $k = 3$ candidate proxies with metrics in the same bin, the probability of collision is upper-bounded by 0.015 (as opposed to a bound of 0.03 for the coarse analysis).

C. Bandwidth Overhead: An Example

For illustration purposes, we now derive an estimate for the bandwidth overhead of our protocol(s). We assume that all messages are either ARP requests or ARP responses. An ARP request or response packet is 28 bytes with a 14 byte ethernet header. Assume a simple linear network model as shown in Figure 4 with the parameters derived in Section V-A, and with negligible collision probability as shown in Section V-B.

![Fig. 4. A Simple Linear Ad-hoc Network Model to Compute Overhead.](image)

It is easy to verify that in the steady state, an invalidation (handoff) occurs at most every second monitoring period. If in the worst case all $d$ destinations are invalidated each time a handoff occurs, and monitoring incurs an overhead of $2k$ ARP messages every 10 seconds, the invalidation incurs an overhead of $6d$ messages every 20 seconds (assuming a per-destination invalidation scheme). The total overhead is, on the average, $6 + 3d$ messages every 10 seconds. At 42 application bytes per message, this translates to an application bandwidth of $0.0252 + 0.126d$ Kbps. In several cases, the legacy only talks to the external world. In this case, $d = 1$, and the application bandwidth overhead is only 0.15 Kbps. Similar estimates can be made for other environments.

VI. PROTOTYPE IMPLEMENTATION

In our prototype, we implemented the wireless ad-hoc node using an off-the-shelf Atheros-based 802.11g wireless...
network interface card (NIC) on a single board computer (SBC) [19] running Linux. The implementation was done at the application layer. Routing packets were injected via the Linux packet socket interface. The Linux iftables and iptables routines were used to intercept routing and protocol packets for processing. For multicast/broadcast data packets that are not part of our protocol, we used a simple packet identification technique to ensure that every node transmitted a packet exactly once. To determine signal strength for received packets, we used the ioctl iwspy interface provided by the madwifi-ng driver. The mean signal strength was computed using 3 measurements. For ad-hoc routing, we used the DSDV [3] table-driven protocol with some enhancements. For the MAC layer, we used standard 802.11 DCF [14] with RTS/CTS turned off.

Our ad-hoc network (see Figure 5) was deployed in a small office environment. The gateway and the DHCP functionalities were supported by node N1. Our legacy devices were DHCP-enabled laptops running Windows XP or Linux (with a 2.6 kernel) with standard 802.11g NICs. Discovery of legacy devices was triggered based on DHCP broadcasts and ARP requests from the legacy device. For installing a route, we used either ARP requests or responses as described earlier in the paper. Keep-alive messages (see Section IV-C) were triggered every 10 seconds to accommodate the mobility arising from an average walker.

Observations. We performed some experiments on our prototype system to determine the performance overheads of the protocol. The experiments were repeated five times and some of the observations are summarized below.

We first located the legacy device at point C in the floorplan shown in Figure 5. In the first set of experiments, we observed a round trip delay of ≈ 20 msecs to receive a DHCP response from the server, which shows that the overhead for propagating messages from the legacy device to/from the rest of the network is small. In the second set of experiments we measured the time that it took for obtaining the routes to a particular destination. During the course of the experiment, the candidate proxy set varied over time. The total round trip delay for an ARP response coming back from the candidate proxy set {N2, N4, N5, N6} was only 2–5 msecs, well within the bounds of ARP timeouts.

In the third set of experiments, we measured the handoff behavior of the legacy device moving 220 ft. from point A to point B in the floorplan at an average walking speed. When at point A, node N5 acted as the proxy for the legacy device. During the walk, the device first handed off to node N6 and then to nodes N2 and N1.

VII. Conclusion

In this paper, we studied the problem of support for legacy devices in a multi-hop ad-hoc network. Our WaterWalker architecture extends the inherent benefits of path redundancy in multi-hop ad-hoc networks to legacy devices. We proposed and analyzed protocols for realizing this support, and also presented an ARP-based implementation scheme for our protocols. Our prototype implementation of the WaterWalker architecture using standard off-the-shelf hardware and wireless interface cards demonstrates the feasibility of our proposals.

REFERENCES