Experiences Using IEEE 802.11b for Service Discovery

Michael S. Thompson, Scott F. Midkiff
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061 USA
email: {stuboy, midkiff}@vt.edu

***Abstract*— This paper shares our experiences developing a service discovery dissemination mechanism for pervasive computing environments. We explore the problems encountered when designing a solution for IEEE 802.11b ad hoc networks, including the performance of multicast and unicast delivery. We show and discuss results from both high and low traffic scenarios, with nodes in close proximity. We also discuss a small test using a multihop configuration.**

Observations in this paper are useful to researchers working in almost any layer of the networking stack who wish to better support multicast and unicast communications in mobile ad hoc environments.

I. Introduction

This work is a portion of a larger effort to build a service discovery solution for pervasive computing environments. The need for service discovery in pervasive computing environments stems from pervasive computing's service-based nature and the need to locate and communicate with services[1], [2]. Pervasive computing must deal with dynamic and static environments using both wired and wireless communication. There are numerous prior works in the area of traditional, wired networks and service discovery[3], [4], [5]. Our work focuses on service discovery in wireless, mobile ad hoc networks (MANETs). In environments of this type, there are many concerns of a service discovery scheme. Of these, we focus on heterogeneous node composition and node populations, both dense and large[6].

Currently, IEEE 802.11b is the most common choice for wireless networking. For our work, we are using IEEE 802.11b in ad hoc mode. Ad hoc mode is a peer-to-peer network architecture. There is no central control point facilitating communication between the nodes in the network, such as an access point. Additionally, nodes may enter and leave the network at their pleasing. Another characteristic of ad hoc networking is the ability to build large, multi-hop networks where neighboring nodes are responsible for forwarding messages from the sending node to the receiving node in the event that the receiving node is out of the radio range of the sending node. We address this scenario briefly. The majority of this work observes single-hop, dense populations of nodes.

Finally, a key element for service discovery and, therefore, pervasive computing, is the ability to locate nodes in unknown environments. To find nodes in an unknown environment, nodes must be able to address and communicate with other nodes. For communication with other nodes, there are three methods of addressing, unicast, multicast and broadcast. Unicast communications include a single sender and a single receiver. Multicast communications employ a single sender transmitting to multiple receivers.[1] Broadcast communications also employ a single sender transmitting to multiple receivers. The difference between multicast and broadcast is broadcast includes the entire set of nodes. Multicast includes a subset of the entire set. By definition, this subset may be the entire set as well. The key is that nodes have control over their involvement in reception of messages. Additionally, multicast may be generalized to include both unicast and broadcast style communications. For this work, we will follow these definitions to avoid confusion.

For location purposes, unicast is of little use since a node must already know the unique address of another node. This is especially true in transient environments where neighboring nodes arrive and leave. To communicate with other nodes, multicast and broadcast are the two choices. Multicast is the method of choice for the following reasons. The lack of broadcast support in some protocols, such as the Internet Protocol version 6 (IPv6)[7]. The lack of broadcast routing in some protocols, such as the Internet Protocol version 4 (IPv4). All nodes are not required to participate; participation is voluntary. Multiple addresses are available, versus a single broadcast address being available (in the case of IPv4).

Our observations of unicast and multicast performance in IEEE 802.11b networks are useful to researchers working with all layers of the network stack above the physical layer. Various key functions, like service discovery, depend on multicast communications. Researchers interested in better supporting these services, can build link, network, transport, and application layer techniques to support the needs of applications and devices, such as lower packet loss rates and more diverse quality of service guarantees.

In Section II we provide a short motivation of this investigation. Results and analysis of the broadcast flooding algorithm are presented in Section III. Results and analysis of a comparison of multicast and unicast performance in IEEE 802.11b networks follow in Section IV. Finally, in Section V,

[1]Using multicast, it is possible that all of the receivers in the group can additionally send to the rest of the group creating a many-to-many scenario.

Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)
0-7695-2520-2/06 $20.00 © 2006 IEEE

we offer observations and discuss future work.

II. Motivation

In peer-to-peer networks, wireless networking is a suitable choice of communication medium. Since IEEE 802.11b is the current *de facto* standard in wireless networking, it is considered in this study. In this paper we analyze the use of IEEE 802.11b for wireless ad hoc networks. Our major concerns are the problems encountered performing service discovery over IEEE 802.11b networks.

In any discovery scenario, a node desires to find at least one other node on the network. Unfortunately, this node may lack previous knowledge of nodes in the network. To communicate with unknown nodes multicast or broadcast must be used. Due to the reasons mentioned in Section I, multicast is the more desirable choice. Current service discovery solutions employing multicast include Universal Plug and Play[3] and Rendezvous[8].

Locating nodes in a mobile ad hoc network is a difficult problem. Service discovery and ad hoc routing are examples of applications that depend on the ability to find and communicate with neighboring nodes. There are two primary ways of locating a node. The first is proactive where surrounding nodes periodically exchange their knowledge of the network. The second is reactive where nodes send out requests for information about the network configuration only when the information is needed. Both of these are plagued by the same problem. They must balance the accuracy of knowledge and resource utilization concerns. More accurate knowledge requires that more packets are sent and analyzed. The resource concerns stem from the use of small devices, which is a constraint we consider.

Due to the nature of radio communication, IEEE 802.11b has a higher packet loss rate than wired networks. In addition to this, most multicast communication takes place over unreliable protocols, such as the User Datagram Protocol (UDP). The lower probability of delivery of multicast packets for discovery protocols makes it more difficult to locate nodes. The simplest technique to increase delivery probability is broadcast flooding[9][2] Although flooding is simple, it is rather resource expensive to use. Here we examine two variations of broadcast flooding.

III. Broadcast Flooding

In the broadcast flooding algorithm, a node transmits a packet. Each node in the group that hears the packet, resends the packet, assuming the node has not already forwarded the packet. The goal of this algorithm is to transmit the packet to every node in the group. The idea is that the probability of a node not seeing at least one of the packets is very small. There are two variations of this scheme. One variation rebroadcasts the packet immediately[9]. The second variation waits a short randomly selected amount of time and then rebroadcasts the packet[9]. The second approach is meant to avoid transmission collisions between neighboring nodes within the network; this is known as the *broadcast storm*[10] problem.

[2]Though it is named the "broadcast" flooding algorithm, we use multicasting for evaluation. This should not affect the performance of the algorithm.

A. Experimental Setup

Our experiment considers a small network of pervasive computing devices that issue queries for services. Our interest is the probability of delivery as the number of nodes in the group increases.

1) Hardware: Our experiments use Hewlett-Packard 3850 iPAQs running Microsoft Pocket PC 2002 with a dual PC card expansion sleeve. These devices include 64 MB of memory and 206-MHz ARM processors. Xircom CWE-1130 IEEE 802.11b PC cards are used for network connectivity.

2) Software: Our application is written in C# using the Microsoft .NET Compact Framework. The application is composed of multiple threads. The sender thread sends packets at randomly chosen intervals between 1 and 10 seconds to a known multicast address and port. The interval differs slightly from packet to packet. The receiver thread listens for packets. Upon reception the packet is checked to see if it is new or not. If the packet is new, the receiver logs the packet information and rebroadcasts the packet. In the case of the random delay variation, the receiver waits between 0 and 200 ms before rebroadcasting the packet. This time was arbitrarily chosen during development, but proves to be a reasonable wait time, as will be seen later. Finally, there are other maintenance threads that handle updating a list of known one-hop neighbors and a list of forwarded packets.

Each packet contains a unique identification string that contains the address of the original sender concatenated with a random number. This information, in addition to the address of the sending (or forwarding) node, a time, and an action ("send", "fwd", or "recv").

3) Setup: The group is assembled into an ad hoc network using 40-bit WEP, automatic channel selection, and automatic address selection. The environment is our research laboratory that has numerous IEEE 802.11b/g access points within range. Measurements were taken across multiple days. Interfering nodes changed depending on the day as different channel coverage was observed on different days. Testing in a noisy environment is of interest because we feel it is closer to reality, as IEEE 802.11 networks are becoming widespread.

Each variation of the flooding algorithm, delay and no delay, is presented. Each consists of tests at 1 mW and 30 mW transmit power with 2- to 6-node groups. Each test was run 10 times and lasted 5 minutes. When using a group larger than six nodes, a socket buffer overflow occurred within the operating system. We have not yet diagnosed this problem.

B. Results and Analysis

This section presents and analyzes the data collected from the tests described above. Figure 1 shows the percentage of delivered packets for 2- to 6-node networks.

In this scenario, broadcast flooding demonstrates that it is not perfect in either variation. This differs from our initial expectations. We would like to investigate larger populations

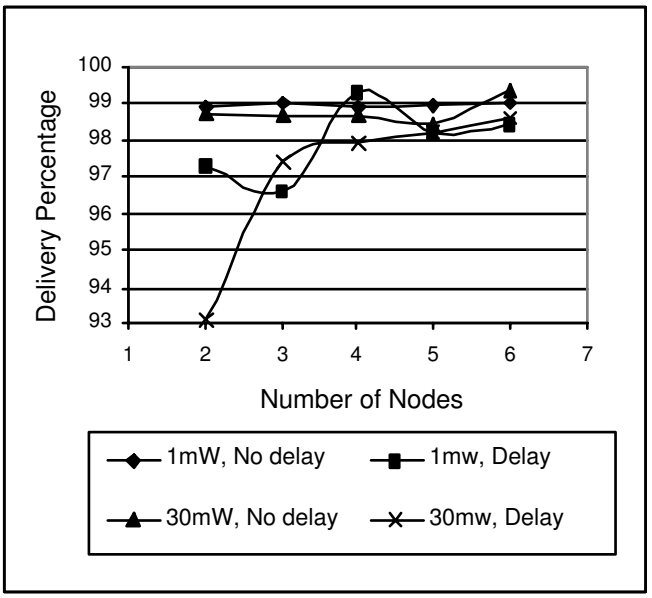

Fig. 1. Delivery percentages for broadcast flooding with close proximity.

to see if this trend continues, as the delay scenarios approach the no-delay scenarios as the population grows. The delay scenarios exhibit a significant deviation with a small number of nodes. Looking at the raw data shows that the 30 mW delay scenario has three consecutive outliers. Ignoring these brings this series back towards the overall trend of this scenario. We attribute these outliers and the variation in the 1 mW delay case to environmental unpredictability.

We also observed a difference in the number of losses between the delay and no-delay scenarios. The no-delay scenarios exhibit the characteristic of missing packets in multiples of $n-1$, where n is the number of nodes within range of the transmitting node. Upon further analysis, we found the missed packet is missed by all nodes within receiving range of the transmitting node. One feasible explanation for this behavior is that receiving nodes may receive and try to retransmit a message while the radio disturbance is still present. The delay scenarios avoid some of this interference by postponing retransmission for a period long enough for the disturbance to subside.

C. Using Unicast

We modified the test application to send unicast packets instead of multicast packets. To avoid making major changes to the code, only 2-node groups were tested. Each node followed the same practices as above, except that it sent packets to the other using its unicast address and not the multicast address. For each of these tests, the delivery percentage was 100 percent; no packets were lost in 5 test runs. This is comparable to the slightly less than 99 percent for the multicast version with 2 nodes.

IEEE 802.11 treats multicast and unicast packets differently[11]. Using Network Instrument's Observer[12], we observed the link layer traffic during the tests. Although, Request To Send (RTS) and Clear To Send (CTS) control messages are available for unicast packets, these mechanisms were not turned on. Since the RTS/CTS mechanism is not available for multicast traffic, the higher delivery percentage was still not explained. By observation, the only difference between the multicast and unicast traffic was use of IEEE 802.11 acknowledgement messages (ACKs) for unicast traffic.

For unicast packets, the IEEE 802.11 MAC protocol, provides a higher assurance of delivery by using acknowledgements for unicast frames[11]. The sender will wait and retry sending the frame until it reaches a set number of retransmissions or an acknowledgement from the receiver is received. This attempts to ensure a higher delivery probability, but comes at a cost, as seen in Section IV

	Run 1	Run 2	Run 3	Average
Unique Sends	417	428	413	419.33
Avg Misses Per Node	132.88	155.77	156	148.22
Avg Miss Percentage	31.86	36.39	37.77	35.34
Avg Node Neighbors	2.77	2.88	3.11	2.92

TABLE I
NINE NODE MULTIHOP EXPERIMENT RESULTS.

D. Multihop

In addition to the close proximity tests, we ran a small number of multihop tests in a larger configuration of nine nodes. The participating nodes were scattered in an office building large enough to separate some nodes from others. The nodes did not move for the three tests. All of the other parameters were the same as the above tests. The transmit power was set to 1 mW and the delayed form of flooding was used.

Table I shows the results for the experiment. The first row is the number of unique packets sent. This is comparable to the number of queries sent. The next row is the average misses per node. This represents the average number of messages that a node never receives. This statistic is much different from those found in other simulations[13]. The average miss percentage is the number of misses divided by the number of unique packets sent. Finally, the average number of neighbor links for each node is presented. Each link represents a neighbor node from which the node heard at some point during the test. There was only one case where a node lost contact and regained contact with a neighbor node. For the rest of the tests, the nodes retained the same neighbor nodes. Connectivity was determined by a node receiving at least one message from the neighbor within a timeout period of 30 seconds. After 30 seconds of not hearing from a node, the first node considers the neighbor node "lost" and removes it from its neighbor list. We note that a node can lose connectivity to another node and frequently miss packets, but stay connected within the timeout period. Making the timeout shorter would expose these nodes, but we do not feel it would make much difference as a disconnection was only experienced once.

There is a large difference in delivery probability compared to the close proximity scenarios. Interesting observations of this data are as follows.

- The number of lost packets tends to increase as connectivity increases.
- The connectivity changed from test to test, but the node placement and environment were static.
- A large number of packets were lost.

At this time, we do not have a good explanation for these results. We are interested in running more tests with different numbers of nodes to try and expose patterns as the number of nodes changes. Also, we would like to try other physical configurations of nodes to observe the difference as connectivity is increased and decreased.

IV. Multicast versus Unicast

Given the differences in unicast versus multicast performance in the above scenarios, we looked further into other possible performance differences. In addition, we wanted to recreate the socket buffer overflow scenario in a different environment. To do this, we built a traffic generation and reception application in an attempt to overflow the socket buffer.

A. Experimental Setup

The traffic generation tool is capable of sending or receiving packets with given parameters. The number of packets, packet size, interpacket delay, and destination information are all available to the user of the generator. On the receiving end, a port and multicast address (if desired) are available for user input. Microsoft C# and the .NET Compact Framework were used for development. The receiver is one of the aforementioned iPAQs. For these tests, a Dell Axim X30 (624-MHz XScale Processor, 64 MB of memory, and built-in IEEE 802.11b) and a notebook computer were used as generators. The Axim was chosen because initial testing showed it was capable of sending packets fast enough to overload the iPAQ.

For our experiments, the packet sizes were 50 byte, and 100 to 1600 bytes in 100-byte intervals. We tested both unicast and multicast packet delivery. For each test, 100 packets were sent. Finally, we varied the delay from 0 ms to 2 ms in 1-ms intervals. Tests were conducted in both our laboratory and a residence. The residence supplied us with an IEEE 802.11b-free zone. We are not sure about other interference in the same band because we do not have equipment to measure this. There was a difference in the delivery performance, so we feel there was considerably less interference in this band compared to our research laboratory. All tests in this set were conducted with nodes in close proximity with a transmission power of 1 mW. Each packet size was tested 5 times.

We have two goals in these experiments. The first is to observe the delivery probability of unicast and multicast, given the same environment. The second is to recreate and resolve the socket buffer overflow observed with the first application.

B. Results and Analysis

Figure 2 shows the unicast and multicast performance for different packets sizes. Using Ethereal, a network protocol analyzer, we observed an average delay of 1.5 ms between packets sent from the Axim. Running the generator on a notebook computer, the interpacket time averaged 150 μs. Using the notebook computer with a delay of 0 ms, the iPAQ received only between 2 to 3 percent of the packets for all packet sizes. Figure 3 shows the performance with an additional millisecond delay. Using Ethereal again, the interpacket delay was about 2.5 ms. At this time we have only collected data for the home scenarios. For delay larger than 1 ms, the delivery percentage rose to almost 100 percent.

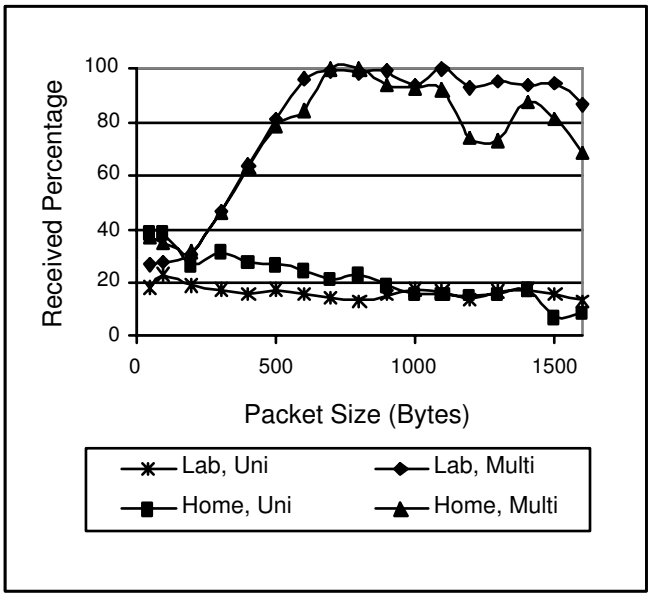

Fig. 2. Percentage of packets received (no delay).

In the scenario with no delay, there is a distinct difference between the multicast and unicast performance. This packet loss happens at the receiver as a notebook computer running Ethereal caught 98 to 100 percent of the packets sent in both unicast and multicast scenarios. For this we assume that the notebook computer with Ethereal, sitting next to the iPAQ, is receiving approximately the same packets that the iPAQ is receiving. Since multicast packets have a higher probability of delivery, the comparatively low amount of processing power of the iPAQ is not wholly the problem. Once again, the only difference between the two scenarios is the use of acknowledgements and retransmissions accompanying the unicast packets within the IEEE 802.11 MAC protocol.

In the 1-ms delay scenario, the unicast and multicast performance are identical until the 400-500 byte range. After this size, the unicast scenario begins to decline in performance. This is from the loss and retransmission of packets. When retransmitting packets larger than 500 bytes, the time it takes to retransmit successfully begins to be longer than the interpacket arrival time. The received percentage of packets larger than

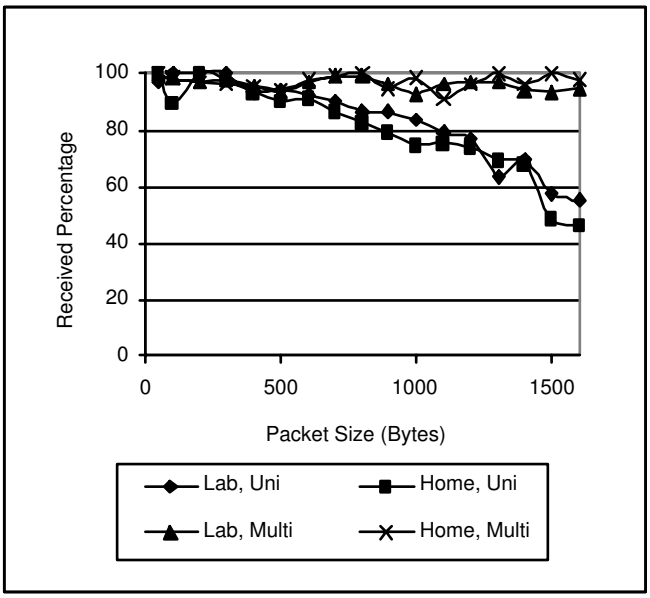

Fig. 3. Percentage of packets received (1 ms delay).

500 bytes in the unicast scenarios continues to decrease as the packet size gets larger. In the multicast scenarios, the received percentage does not drop below 90 percent, even as the packet size increases.

In both cases, there is a large performance degradation for packet sizes approaching 1500 bytes. Packets larger than 1470 bytes are fragmented by the network layer before being transmitted, meaning two packets must be transmitted instead of one. We included these packet sizes for general interest.

Unfortunately, we could not recreate the socket buffer overflow using the traffic generation tool on the Axim or the notebook computer. We are currently unsure of what causes this error, but will attempt to diagnose it in the future.

V. Conclusion

In this paper, we addressed three topics. The first topic was observation of IEEE 802.11b in ad hoc mode for nodes in close proximity and low volume traffic scenarios using multicast delivery. A small number of distributed proximity, multihop tests were included, as well. These scenarios are useful in analyzing the problems encountered by information dissemination schemes being used in wireless ad hoc environments, such as service discovery. We then compared the performance of multicast delivery to unicast delivery, demonstrating the differing performance of IEEE 802.11b in each mode.

The second portion of the paper observed nodes in close proximity and high volume traffic scenarios of with IEEE 802.11b in ad hoc mode. The main goal was to compare unicast and multicast performance given different packet sizes. While this pertains less to information dissemination, it exposes the difference in performance for unicast and multicast using IEEE 802.11b.

The final portion of the paper explored problems we encountered in building the tools for the tests.

VI. Future Work

We have two other scenarios where we would like to test IEEE 802.11b behavior, a sparsely distributed multihop environment and a hybrid environment with sparse distributions of dense clusters with multiple hops between nodes in one group to nodes in another group. These three scenarios comprise a majority of the scenarios where service discovery may be used. Finally, these characteristics will be used to design a service discovery dissemination scheme.

Outside of service discovery, an area for future work is the design of a wireless communication protocol that gives better or equal support to multicast traffic. This is contingent upon the use of multicast in non-high speed streaming applications, as considered in this paper.

Acknowledgement

Michael Thompson is partially support by a National Science Foundation IGERT grant, award number DGE-9987586. This work was partially supported by a grant from Microsoft Research. We would also like to thank Mary Thompson for help in the collection of data, Joshua Edmison for help with code debugging, and Stewart Griffin for comic relief.

References

[1] M. Weiser, "The computer of the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, Sept. 1991.

[2] M. S. Thompson and S. F. Midkiff, "Service discovery: A design framework and survey of current solutions," 2005, unpublished manuscript.

[3] (2003) UPnP device architecture 1.0. UPnP Forum. [Online]. Available: http://www.upnp.org/resources/documents.asp

[4] (1999, June) Salutation architecture specification v2.0c, parts 1-3. Salutation Consortium. [Online]. Available: http://www.salutation.org/specordr.htm

[5] E. Guttman, C. Perkins, and J. Veizades. (1999, June) RFC 2608: Service location protocol, version 2. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2608.txt

[6] M. S. Thompson and S. F. Midkiff, "Service description for pervasive service discovery," in *The First International Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI'05)*, June 2005, pp. 273–279.

[7] S. Deering and R. Hinden. (1998, Dec.) RFC 2460: Internet protocol version 6 (IPv6) specification. IETF. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt

[8] (2004, Mar.) Rendezvous network services architecture. Apple Computer. [Online]. Available: http://developer.apple.com/documentation/Cocoa/Conceptual/NetServices

[9] K. Obraczka, K. Viswanath, and G. Tsudik, "Flooding for reliable multicast in multi-hop ad hoc networks," *Wireless Networks*, vol. 7, no. 6, pp. 627–634, 2001.

[10] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *MobiCom '99: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 151–162.

[11] B. OĤara and A. Petrick, *IEEE 802.11 Handbook: A Designerś Companion*, 2nd ed. IEEE Press, 2005.

[12] Observer. Network Instruments. [Online]. Available: http://www.networkinstruments.com/products/observer.html

[13] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A performance comparison study of ad hoc wireless multicast protocols," in *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, vol. 2. IEEE, 2000, pp. 565–574.