Experimental Evaluation of a Wireless Ad Hoc Network

Saman Desilva
Division of Computer Science
The University of Texas at San Antonio
San Antonio, TX 78249-0667
U.S.A.
sdesilva@cs.utsa.edu

Samir R. Das
Department of Electrical & Computer Engineering and Computer Science
University of Cincinnati
Cincinnati, OH 45221-0030
U.S.A.
sdas@ececs.uc.edu

Abstract

We experimentally evaluate the performance of a wireless ad hoc network from the point of view of both the routing and transport layers. The experiments are done on a testbed with desktop PCs and laptops using wireless radio LAN interfaces. For these experiments an on-demand routing protocol called AODV has been implemented as a part of the operating systems protocol stack. We describe our design choices and the experimental setup. The performance evaluation reveals that the performance is poor beyond two hops at moderate to high loads.

1. INTRODUCTION

Recent advances in wireless communication technology and portable computing devices such as notebooks, palmtops and PDAs have generated a lot of interest in infrastructure-less or *ad hoc* networking. The idea here is to network a collection of portable computing devices using low-cost, short-range radios without any static infrastructure such as base stations or access points. In an ad hoc network each node act as a potential router, routing packets for two communicating nodes that may not be radio contact with each other. Thus communication may be via multiple wireless hops. There are a couple of technical challenges that must be addressed to make such networks usable in practice. First, as the nodes can be mobile and thus the topology can be changing, a dynamic routing protocol must be employed to maintain routes between a pair of source-destination nodes. Second, the access to the shared wireless medium by a number of competing nodes must be efficient and fair. The nature of the wireless medium makes the medium-access control (MAC) problem nontrivial. For example, the received power and hence the SNR (signal-to-noise ratio) falls rapidly with the distance of a receiving node from the transmitting node. Thus, it is not ordinarily possible for the transmitter to sense the carrier or detect packet collision at the receiver.

Conceivably, the effectiveness of both the routing and MAC protocol will affect the performance observed at the higher layers (e.g., transport and application layers). Recently, progress has been made in studying both routing and MAC protocols in ad hoc networks. In particular, a new class of routing protocols has been developed that maintain routes in an "on-demand" or "as needed" fashion [3][7]. On-demand protocols use a source-initiated route discovery process to discover routes when the route is needed. When links in a route being used break due to node mobility, for example, part of that route becomes stale. The stale part of the route is erased and a new route discovery is initiated. Depending on the protocol and scenario, the broken route may be locally repaired as well. On-demand protocols are attractive compared to more traditional, proactive, shortest-path based protocols (e.g., those based on distributed Bellman-Ford [2] or distributed link-state protocols [19]), as they usually have a lower routing overhead for common traffic scenarios [20]. Currently, several on-demand protocols are under consideration for standardization in the MANET (mobile ad hoc networking) working group [15] in the IETF (Internet Engineering Task Force).

Progress has also been made in designing random access MAC protocols based on carrier sense multiple access (CSMA). For example, the IEEE standard 802.11 [13] specifies physical and MAC layer protocols for wireless LANs. It uses a CSMA/CA (carrier-sense multiple access with collision avoidance) protocol that persists on a busy channel and employs a random backoff after the channel switches to idle to avoid the possibility of collision at the receiver. The 802.11 standard optionally uses RTS/CTS (request-to-send and clear-to-send) packets to reserve the channel between a pair of source and destination to avoid the classical hidden-terminal problem [8]. A body of recent work evaluates the performance of the 802.11 standard in an ad hoc setting (see, for example, [12]).

Most of the performance study on routing and MAC-layer protocols in ad hoc networks has been on simulation models (see, for example,[14]). However, even the most detailed simulation models available do not model various limitations of a real wireless network. These include limitations of the interface hardware such as limited buffer space, limitations due to operating systems protocol stack implementation and/or slow processor such as slow memory buffer allocation, or various random sources of errors in the wireless physical layer, such as multipath propagation, radio noise, or occasional radio shadow owing to mobility, asymmetric radio links etc. In spite of these limitations of simulation studies, only very limited amount of work has appeared in current literature evaluating the performance of real multihop wireless networks [6].

In the following, we describe our experience in the design and evaluation of a wireless ad hoc network using Linux laptops and Lucent Technology's WaveLan radio [1]. We have implemented an ad hoc network routing protocol called AODV (***A****d hoc* ***O****n-demand* ***D****istance* ***V****ector*) routing

[3][4] as a part of the Linux network protocol stack. The protocol allows setting up and maintenance of routes from a given source to any destination(s). The WaveLan radio uses a CSMA/CA protocol for access to the radio medium. We have used the first-generation WaveLan radios (circa 1997) that do not have an implementation of the RTS/CTS based channel reservation technique and also do not have any ACK following a successful packet reception. Our intention is to experimentally evaluate the performance of the AODV routing protocol (e.g., route discovery latency) at the network layer and UDP/IP and TCP/IP protocols at the transport layer on this testbed.

The rest of the paper is organized as follows. Section 2 describes our testbed and presents results from a set of single hop experiments that also demonstrates the hardware characteristics. Section 3 describes the AODV protocol implementation in the Linux kernel. Section 4 presents the performance evaluation experiments, followed by conclusions in section 5.

2. WIRELESS AD HOC NETWORK TESTBED

The wireless testbed consists of one 500 MHz, 128 MB Pentium desktop PC and four 120 MHz, 24 MB Pentium laptops (IBM ThinkPads), and one 400 MHz, 128 MB Pentium laptop (Dell Inspiron). Each machine runs the Linux Operating System Version 2.2.12. The wireless interfaces are Lucent Technology's WaveLan [1] ISA (for desktop) and PCMCIA (for laptop) cards. Each card contains a LAN controller, modem control unit and a radio transceiver and is attached to a small external unit that houses the antenna. The WaveLan system used operates in the 2.419–2.445 GHz license-free ISM (industrial-scientific-military) band at a nominal bit rate of 2 Mbits/sec. WaveLan employs a low-power radio (transmit power about 500 milliwatts) and is primarily targeted for in-building wireless extensions of an existing Ethernet LAN. We, however, used WaveLan in an ad hoc setting.

We performed extensive performance evaluation of a single one-hop wireless link for both PCMCIA and ISA cards. This is because previously reported evaluations with WaveLan showed that the architectural differences in these cards and their drivers as well as processor differences between a desktop and a laptop may affect the network performance significantly [11]. The key results from this evaluation are summarized in Table 1. The performance measurements on either type of laptop (400 MHz and 120 MHz) revealed similar results. We found that the PCMCIA card has a maximum send bandwidth 1.2 Mb/s, which is much less than the nominal. The ISA card, however, can go up to 1.8 Mb/s. This limitation of the send data rate on PCMCIA is due to a single buffer allocation and high buffer transfer time from the device driver to the PCMCIA device. See Table 1 for more details.

Metrics evaluated	PCMCIA	ISA
Maximum send bandwidth (for UDP transfer)	1.2 Mb/s	1.8 Mb/s
Maximum received bandwidth (for UDP transfer)	1.8 Mb/s	1.8 Mb/s
Number of MTU size buffers allocated for data transfer from device to kernel	5	31
Number of MTU size buffers allocated for data transfer from kernel to device	1	10
Time to move data from device to kernel (MTU/RREQ)	2.057ms/ 0.139ms	0.955ms 0.048ms
Time to move data from kernel to device (MTU/RREQ)	3.46 ms / 1.13 ms	0.96ms/ 0.067ms
Time to complete data transmit by device (MTU/RREQ)	6.26 ms / 0.47 ms	6.33ms/ 0.571ms

Table 1. Single hop performance of PCMCIA and ISA WaveLan network cards. The Maximum Transmission Unit (MTU) is 1500 bytes. An RREQ (to be described in the next section) packet is 68 bytes.

3. AN IMPLEMENTATION OF THE AODV ROUTING PROTOCOL

In this section we describe our implementation of the AODV protocol as a part of the Linux protocol stack. We start with a brief description of the protocol itself and then go over to the implementation details.

3.1. AODV Routing Protocol

AODV [3][4] maintains a routing table (essentially, <*destination node, next hop, no. of hops to destination*> tuples) on each node in the ad hoc network. When a node attempts to send a data packet to a destination for which it does not already know the route (i.e., does not have a routing table entry), it uses a "route discovery" process to dynamically determine such a route. Route discovery works by flooding the network with route request (RREQ) packets. Each node receiving a RREQ, rebroadcasts it, unless it is the destination or it has a route to the destination in its routing table. Such a node replies to the RREQ with a route reply (RREP) packet that is routed back to the original source. Routing table entries pointing back to the source is used to route the RREP back to the source. These entries (called *reverse path* entries) are created at the time RREQ is forwarded. The reverse path entries are expired after a short interval of time only sufficient to allow RREP to be propagated.

If any link on a source route is broken, the source node is notified using a route error (RERR) packet. The source and any intermediate node on the way of the RERR packet remove the indicated route from their routing tables. The RERR propagation works in the following fashion. A set of

predecessor nodes is maintained for each routing table entry on every node. They indicate the set of neighboring nodes that use that entry to route data packets. These nodes are notified with RERR packets when the next hop link breaks. Each predecessor node, in turn, forwards the RERR to its own set of predecessors, thus effectively erasing all routes using the broken link.

An important feature of AODV is the maintenance of timer-based states in each node, regarding utilization of individual routing table entries. A routing table entry is "expired" if not used recently. Only useful routes are maintained to keep routing overheads low. AODV also uses a sequence number-based technique to determine freshness of routing table entries. The sequence numbers work like logical clocks. They also help in guaranteeing loop freedom in the protocol. More details of AODV protocol activities and necessary data structures can be found in [4].

3.2. Design Choices

We implemented the AODV protocol as an extension to the Address Resolution Protocol (ARP) [10][21]. ARP is useful in a broadcast network like Ethernet to provide an IP-to-MAC address mapping so that packets in a broadcast domain can carry an identifying MAC address corresponding to the intended destination. ARP maintains an ARP table that contains all known mappings. If a mapping is absent in a node S, node S does an ARP request using a MAC broadcast packet carrying the IP address of the intended destination D. Node D being in the same broadcast domain hears this request, recognizes the address to be its own, and responds to S with an ARP reply thus supplying its MAC address. S now can insert the new mapping learnt in its ARP table. A technique called Proxy ARP [10][21] can be used when S and D are in different broadcast domains but there is no IP-layer router to route IP packets from S to D. In proxy ARP, a "gateway" node N connected to both the broadcast domains responds to the ARP request for node D from node S. Node S now maps the MAC address of node N for the IP address of node D in its ARP table. Thus, all IP packets for node D are now targeted for node N. Node N now can use the usual IP-routing approach (assuming, of course, that the two broadcast domains are appropriately "subnetted") to forward the packets to D.

This Proxy ARP approach can be extended with minor modifications in the ARP layer to handle multi-hop forwarding of IP packets. In our model of the ad hoc network, all nodes belong to the same IP subnet and "appear" to the IP layer to belong to the same broadcast domain as if they are on the same Ethernet segment. An extension of the above proxy ARP approach is used for multihop routing. The ARP table essentially acts as the AODV routing table by providing a next hop MAC address for each destination IP address. Each node that is not the destination of an IP packet, forwards the packet to this next hop MAC address by consulting the ARP table. If the source node (as specified in the IP header) does not have an entry corresponding to the destination address in its ARP table, it starts a route discovery process by broadcasting a modified ARP request packet, which now serves as the AODV RREQ packet. Any node receiving this modified ARP request packet for the first time rebroadcasts it, unless it is the destination or has an entry for the destination in its ARP table. In the latter cases, the node generates a modified ARP reply (now AODV RREP) targeted for the IP address of the source node and forwards it to the MAC address of the immediate source of the ARP request. Each ARP request received for the first time also updates the ARP table thus providing a reverse multi-hop route back to the original source.

The above mechanism using proxy ARP is very efficient and requires only minimal modifications of the ARP packet structure and ARP table to include AODV specific fields.

3.3. Local Connectivity Management

In the absence of any link-layer ACKs (recall that the radios we have used do not have them) the AODV protocol needs to keep track of the local neighborhood explicitly. Otherwise, there will be no way of knowing when an existing link is broken. Each time a neighbor moves out of the neighborhood, the protocol will invalidate all routes through that neighbor. To keep track of the neighborhood, each node periodically (every second in our testbed) broadcasts a *hello* message. Whenever a node receives a *hello* message form a neighbor, it checks and possibly updates it local connectivity information that is maintained as a set of neighboring nodes. When a node does not hear from an existing neighbor for predetermined period of time (2 seconds in our testbed), it assumes that the link to the neighbor is lost. In this case, the set of neighbors is updated and an error mechanism via RERR is initiated.

3.4. Route Errors and Queuing

RERR packets are ARP reply packets with a special flag to differentiate them from RREP packets. In the current specification of AODV, an intermediate node does not start any new route discovery. This is the responsibility of only the source node. When an intermediate node has no route to a destination (i.e., no ARP table entry corresponding to the destination IP address of an incoming packet, whose source IP address is different from its own), it will drop all incoming packets for that destination. On the other hand, the source node will start a new route discovery and will buffer the IP packet until a route is found.

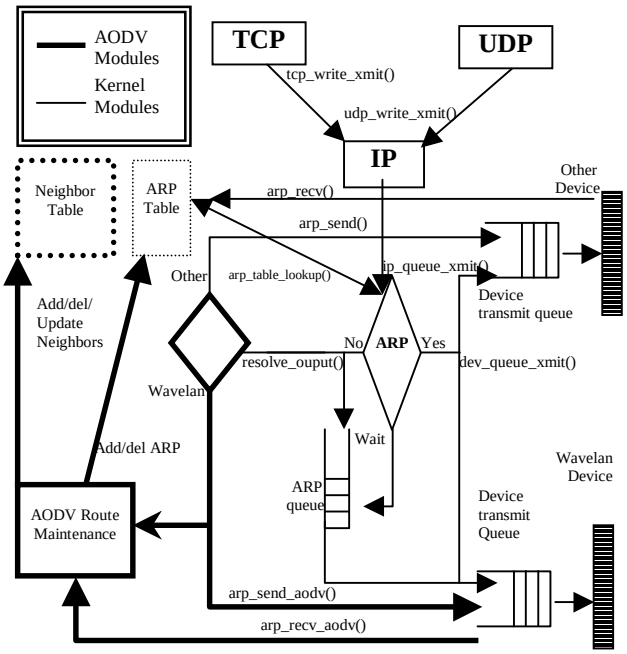

Figure 1. Block diagram of AODV

The Linux kernel maintains two queues that are useful in this context. One queue buffers the IP packets waiting for an ARP reply (called *ARP queue*) and another buffers packets that received a MAC address from the ARP table, but are just waiting to be transmitted by the device driver (called *transmit queue*). See Figure 1 for a diagram. In Linux, the transmit queue does not have any bound. But the ARP queue can hold only 3 packets. Since this short queue is not sufficient for an on-demand routing protocol, in our experiments the ARP queue size was increased to 30.

4. EXPERIMENTAL EVALUATION

The keep the experiments manageable and repeatable they were run in a static (non-mobile) configuration. In some experiments route breaks are emulated by artificially purging the routing table (ARP table) entries using an independent daemon program to force new route discoveries. In others, route loss was a part of the natural dynamics of the network because of loss of hello messages.

4.1. AODV Route Discovery Latency

The route discovery latency is the time to discover routes. It is a key performance metric for on-demand routing protocols like AODV. Larger latency impacts end-to-end delay of data packets, as packets are buffered while route discovery is in progress. Also with very high route discovery latency, packets can be dropped due to buffer overflows. To determine the route discovery latencies an experiment was run with five nodes set up in a linear chain. The distances between the nodes are adjusted such that a node cannot hear any other node except its neighbor(s) in the chain. The first node in the chain is a desktop with ISA card. The rest all are laptops with PCMCIA cards.

In the first experiment routes are broken artificially by purging routing table entries. There is no traffic in the network except routing packets. Table 2 shows the average and confidence intervals for the latencies for 300 route discoveries for various routes lengths. Single hop route discovery latency is very short (less than 3ms), but each additional hop takes roughly 4ms. The short one hop discovery latency is an artifact of our experimental design. Recall that the first node is a desktop with an ISA card. The primary reason of low single hop route discovery latency is because the ISA card has a much lower transfer time between device and kernel compared to the PCMCIA card (see Table 1.) Note again that in our set up only the first (source) node has an ISA card, the rest are all PCMCIA cards. An independent set of measurements verifies that with a laptop used as a source the single hop route discovery time is also about 4 ms.

No. of hops	Average Latency (in ms)	95% confidence interval as a fraction of average
1	2.73	5.73 %
2	6.63	10.47 %
3	10.13	0.74 %
4	14.14	5.07 %

Table 2. Average route discovery latencies for various numbers of hops (with no load)

No. of hops	Average latency (in ms)	95% confidence interval as a fraction of average
1	2.94	49 %
2	15.4	38 %
3	476	31 %
4	700	36 %

Table 3. Average route discovery latencies for various numbers of hops (with multihop UDP).

The above experiment was run without any traffic, but with the first node sending UDP packets (MTU size, 1472 bytes plus IP header) to the destination node at the end of the chain at 0.6 Mb/s, route losses became natural because of loss of hello messages due to collisions etc. So routes no longer needed to be broken artificially. Notice that with a loaded network, the latencies are much higher and also have significantly wider confidence intervals denoting a lot of variability in the data (see Table 3). Notice also the sharp rise in latency with hop counts. This is due to frequent timeouts owing to the loss of RREQ or RREP packets.

4.2. Multihop UDP

We now repeat similar experiments with UDP with nodes in a linear chain, but focus instead on the UDP performance. As before the first node (desktop with ISA card) acts as the UDP source and loads the network with UDP packets destined for the last node in the chain. On each hop, number of packets received, number of packet forwarded, and number of packets dropped/lost for various reasons were recorded via extensive instrumentations. Experiments were repeated for various offered load (i.e., sending rate) and different number of nodes in the chain. As before the packet size is kept fixed at MTU. Each experiment is run for 300 sec and the average of several runs were used.

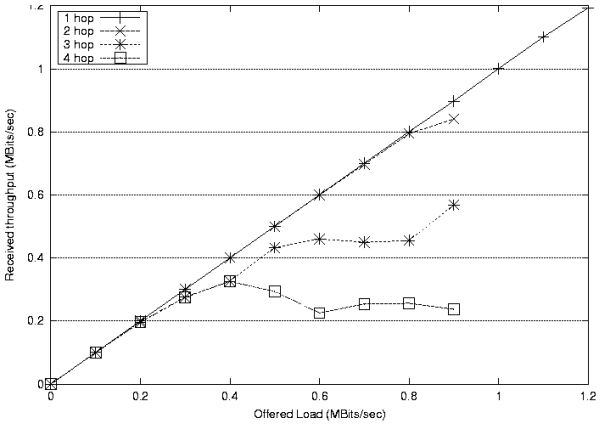

Figure 2. Offered load vs. Received throughput for the multi-hop UDP experiment

Results from above experiment are shown in Figures 2, and Tables 4 and 5. Figure 2 shows offered load and received throughput for 1, 2, 3 and 4 hop experiments. In 1 and 2 hop experiments the received throughput is almost ideal, but the throughput drops considerably for the 3 and 4 hop experiments for higher offered loads. To analyze the reason behind this dropped throughput, we show the packet loss statistics for the 3 and 4 hop experiments in Tables 4 and 5, respectively. In these tables "node *x* % drops" are percentage drops from the routing layer at node *x* with respect to the total number of packets sent and "node *x – x+1* % drops" are the percentage drops from the link layer between node *x* and node *x+1*. For all hop counts maximum error free load appears to be 0.2 Mb/s. Although higher bandwidths are achievable, loss rates are significant for these bandwidths.

In the 3 hop (i.e., 4 nodes) experiment (Table 4), most losses are in nodes 1 and 2 and in between. Here, nodes and 1 and 3 suffer from hidden terminal problem [8]. Both send UDP packets, but are not within hearing range from each other. The packets thus collide at node 2, which loses both UDP packets from node 1 (node 1-2 drops) and hello messages from node 3. Similarly, node 3 loses UDP packets from node 2 (node 2-3 drops) for collisions with the hello messages from node 4. When the load is high, successive losses of hello messages are frequent. Two or more successive losses of hello messages will initiate a RERR message. All UDP packets are then dropped (node 1 and node 2 drops) until a new RREQ is initiated. Even after a new RREQ is started UDP packets could be dropped if the ARP queue runs over. These are also counted in node 1 drops. Notice that packet drops become noticeable beyond 0.2 Mb/s offered load and becomes significant beyond 0.4 Mb/s offered load. The maximum received bandwidth is only about 0.6 Mb/s. Also notice that beyond node 3 packet drops are small. This is because drops in the first two nodes and links are so high, only a small fraction of the offered load reaches node 3 and beyond.

Offered load (Mb/s)	Node 1 % drops	Node 1-2 % drops	Node 2 % drops	Node 2-3 % drops	Node 3 % drops	Node 3-4 % drops	Received %
0.1	0.00	0.00	0.00	0.00	0.00	0.00	100.00
0.2	0.00	2.12	0.00	0.18	0.00	0.22	97.49
0.3	0.64	7.28	0.05	0.14	0.00	0.21	91.67
0.4	0.00	18.02	0.04	0.29	0.07	0.08	81.50
0.5	1.56	10.07	1.25	0.24	0.00	0.40	86.48
0.6	0.00	22.70	0.03	0.53	0.03	0.03	76.67
0.7	3.34	10.27	7.77	14.28	0.00	0.11	64.22
0.8	8.90	10.67	5.25	18.24	0.02	0.08	56.84
0.9	2.63	12.30	9.79	12.12	0.00	0.13	63.04

Table 4. Packets dropped as a percentage of the total number of packets sent (in 3 hop UDP experiment).

Offered load (Mb/s)	Node n % drop								Received %
	1	1-2	2	2-3	3	3-4	4	4-5	
0.1	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.00	100
0.2	0.43	0.3	0.0	0.2	0.0	0.5	0.0	0.27	98.23
0.3	1.6	2.0	0.0	2.7	0.0	1.5	0.0	0.24	91.70
0.4	5.3	5.6	0.7	3.8	0.0	3.2	0.0	0.04	81.16
0.5	14.	6.3	7.7	13	0.0	.19	0.0	0.02	58.49
0.6	25	3.2	13	19	0.0	.14	0.0	0.04	37.47
0.7	15	4.4	11	31	0.0	.14	0.0	0.03	36.17
0.8	18	0.4	7.0	42	0.0	.09	0.0	0.02	31.94
0.9	2.3	22	6.9	42	0.0	.03	0.0	0.05	26.28

Table 5. Packets dropped as a percentage of the total number of packets sent (in 4 hop UDP experiment).

In the 4 hop experiment (Table 5) the situation is similar. But here node 4 also transmits UDP packets, thus increasing the possibility of collisions at node 3 with packets from node 2. Thus, node 2-3 drops are now more significant than before. The maximum received bandwidth is now less, about 0.4 Mb/s. Also notice that received bandwidth is going down with very high load. This is because of a larger number of

drops at node 1 due to ARP queue overflows. With larger number of hops, the route discovery now takes more time (seen in the previous section) thus opening up the possibility of more losses.

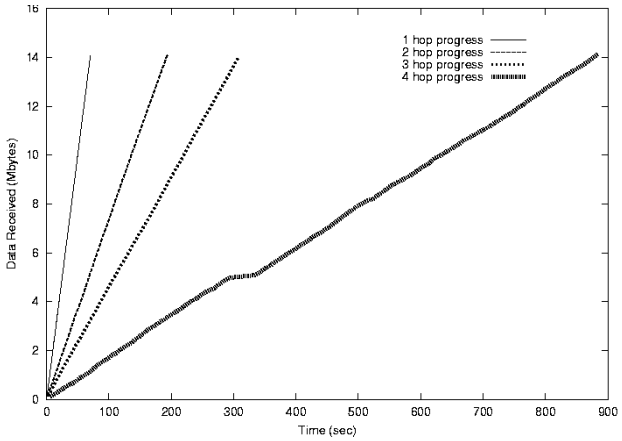

Figure 3. FTP Progress with time for various number of hops. The received bandwidth is 210, 70, 47, 18 KBytes/s (for hop 1, 2, 3 and 4, respectively)

4.3. TCP Performance

A linear chain of nodes was also set up to evaluate TCP performance of multi hop wireless network. This is a set up similar to that in the previous sections, except that the nodes were carefully positioned in such as a way that nodes two hops away can still sense each other's carrier, i.e. the signal strength at a node two hop away is strong enough that the nodes will back off each other via the CSMA/CA MAC protocol (and thus no longer are hidden from each other), but weak enough that they will not form a link. This improves performance somewhat by reducing collisions. Without this "adjustment," we have had difficulty in running FTP with large number of nodes, as the transfer would hang after some time due to a large number of lost packets.

A single FTP file transfer (size = 14 Mbytes) was set up from the desktop (node 1) to the last node in the chain for various numbers of nodes in the chain giving us data for various hop counts. For each experiment the *tcpdump* [21] tool was run on the source node to trace all outgoing and incoming TCP packets. *Tcpdump* is a user-level program that captures link layer packets. For all experiments in this section *tcpdump* was run in non-promiscuous mode minimize load and to avoid capturing snooped packets destined for other nodes. An independent experiment was run to verify that the desktop is powerful enough that *tcpdump* does not steal enough cycles to alter TCP performance in any significant extent.

Figure 3 indicates the progress of the FTP transfer at the destination node for various hop counts. As the hop count increases the transfer gets progressively slower. This is expected from our experience with UDP in the previous section. Even though the collisions and packet drops have reduced somewhat with our new node placement, latency at each node has increased with increased backoff delays.

To further evaluate the reason for slowdown from the perspective of TCP dynamics, we zoom into a randomly selected area of the plot (2 MB – 2.025 MB) and show each individual TCP packets sent (data packets) and received (ACK packets) at the source (see Figure 6). Linux uses TCP Reno [9][18] with selected acknowledgement (SACK) option [17]. Notice the smooth behavior for the data and ACK packets for the 1-hop experiment. In TCP Reno normally alternate packets are acknowledged when there is no packet loss. This is clearly evident from this figure. However, as the number of hops is increased things turn somewhat irregular. For example, with 2 hops the data packets are not sent at as regular an interval as before. There is a small delay after 2 or 4 packets due to TCP window overruns. This means that the TCP sender is waiting for an ACK to move its window or timer expiry for a retransmission. This delay progressively increases with 3 and 4 hops. With 4 hops the situation is bad enough that several retransmissions are observed in the plot. Also notice that the ACKs are not coming back as regularly for the 2, 3 and 4 hop cases, with things getting progressively worse for larger number of hops. Often more than two segments are acknowledged cumulatively by a single ACK. At other times every successive segment is acknowledged. In the 4 hop experiment, several duplicate ACKs are seen. Duplicate ACKs and retransmitted data segments mean dropped packets in the network.

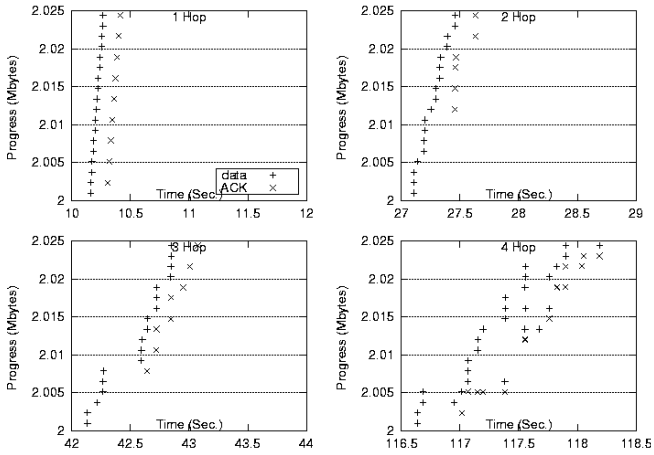

Figure 4. FTP Progress (blowup of Figure 3)

5. CONCLUSIONS

Our work is one of very few that attempts to experimentally evaluate the performance of a wireless ad hoc network by building a dynamic routing protocol as a part of the protocol stack. The multi-hop performance of a wireless ad hoc network has not been very encouraging in our evaluation. Beyond two hops only a small fraction of the nominal bandwidth is actually achievable even for a sparse

network configuration like a linear chain. Packet loss rate is very high unless the load is kept substantially low. We also found that the multiple-access interference (e.g., collisions due to hidden terminals) interacts with the routing protocol (e.g., loss of hello messages or routing packets) significantly at high loads. This gives rise to unnecessary route discoveries even in the absence of mobility. At high load, route discovery latency can also be quite large with high degree of variability, which affects the performance of higher layer protocols such as UDP and TCP due to long packet delays and lost packets due to buffer overflows. But at low load both the routing layer performance (small route discovery latency) and transport layer performance (lower drops) are reasonable.

However, these experiments were done using the first generation wireless LAN hardware. The new generation hardware carries the 802.11 standard and implements link-layer acknowledgements as well as channel reservation using RTS/CTS control packets. This will reduce the impact of the multiple-access interference somewhat. With link layer acknowledgements AODV will also be able to do without the hello messages. In our future work, we plan to use newer wireless hardware with improved device driver as well as optimized implementations of AODV and study performance for realistic applications.

ACKNOWLEDGMENT

The work presented in this paper is partly supported by DoD/AFOSR grant F49260-96-1-0472, and NSF grants ACI-9733836 and ANI-9973147.

REFERENCES

[1] Bruce Tuch, "Development of WaveLAN, An ISM Band Wireless LAN", *AT&T Technical Journal*, Vol 72, Page 27-33, July/Aug 1993.

[2] C. Cheng, R. Riley and S.P.R Kumar, "A loop-free extended bellman-ford routing protocol without bouncing effect", *Proceeding of the 1989 ACM SIGCOMM Conference*, Pages 224-236, 1989.

[3] C. E. Perkins and E. M. Royer, "Ad-Hoc On-Demand Distance Vector Routing", *Proceeding of the IEEE workshop on Mobile Computing Systems and Applications*, page 90-100, February 1999.

[4] C. E. Perkins, E. Royer and S. R. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing," *Internet Draft (work in progress)*, http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-05.txt, March 2000.

[5] C. L. Fullmer and J. J. Garcia-Luna-Aceves, " Solutions to hidden terminal problems in wireless networks", *Proceedings of the ACM SIGCOMM '97 conference*, pages 39 – 49.

[6] D. A. Maltz, J. Broch and D. Johnson, " Experiences Designing and Building a Multi-hop Wireless Ad Hoc Network Testbed", CMU Technical Report CMU-CS-99-116, March 1999.

[7] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks", in *Mobile Computing*, Kluwer Academic, 1996.

[8] F. A. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part-II – the hidden terminal problem in carrier sense multiple-access models and the busy tone solution", *IEEE Transactions in Communications*, pages 1417-1433, 1975.

[9] F. Anjum and L. Tassiulas, "On the Behavior of Different TCP Algorithms over a wireless Channel", *Proceeding of the SIGMETRICS'99 Conference*, Pages 155-165, May 1999.

[10] G. Wright and W. R. Stevens, "*TCP/IP Illustrated. Vol 11"*, Addison Wesley, 1995.

[11] G. Xylomenos and G. C. Polyzos, "TCP and UDP Performance over a Wireless LAN", *Proceedings of the IEEE INFOCOM 99 Conference*, 1999.

[12] H. S. Chhaya and S. Gupta, "Performance Modeling of Asynchronous Data Transfer Methods of IEEE 802.11 MAC Protocol", *Proceedings of IEEE Personal Communications Conference*, pages 8-15, October 1996.

[13] IEEE Standards Department, IEEE 802.11 standard for wireless LAN, medium access control (MAC) and physical layer (PHY) specifications, 1997.

[14] J. Broch , D. A. Maltz , D. B. Johnson, Y-C. Hu and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", *Proceedings of the 4th International Conference on Mobile Computing and Networking (ACM MOBICOM'98)*, pages 85-97, Oct. 1998.

[15] J. Marker and S. Corson, "Mobile Ad hoc networks (MANET)," http://www.ietf.org/ html.charters/manet-charter.html, IETF MANET Working Group Charter, 1997.

[16] M. Gerla, R. Bagrodia, L., Zhang, K. Tang and L. Wang, "TCP over Wireless Multi-hop Protocols: Simulation and Experiments", *Proceedings of IEEE ICC'99*, June 1999.

[17] M. Mathis, J Mahdavi, S. Floyd, A. Romanow, RFC 2018, "TCP Selective Acknowledgment Option", available from http:/www.ietf.org/rfc, Oct 1996

[18] S. Floyd, T. Henderson, RFC 2582, "The New Reno Modification to TCP's Fast recovery Algorithm", available form http:/www.ietf.org/rfc, April 1999

[19] S. Keshav. "An Engineering Approach to Computer Networkings: ATM Networks, the Internet, and the Telephone Network," Chapter 11, Addison-Wesley, 1997.

[20] S. R. Das, R. Castaneda, J. Yan and R. Sengupta, "Comparative Performance Evaluation of Routing Protocols for Mobile, Ad hoc Networks," *Proceedings of the 7th Int. Conf. on Computer Communications and Networks (IC3N)*, Lafayette, LA, pages 153-161, October, 1998.

[21] W. R. Stevens, *TCP/IP Illustrated. Vol I.* Addison Wesley, 1994.