Experiment and Evaluation of a Mobile Ad Hoc Network with AODV Routing Protocol

Kalyan Kalepu, Shiv Mehra and Chansu Yu,
Department of Electrical and Computer Engineering
Cleveland State University
2121 Euclid Avenue, SH 332, Cleveland, OH 44115

Abstract

An ad-hoc network is a collection of mobile nodes forming a network in which the network topology changes dynamically. The nodes use the service of other nodes in the network to transmit packets to destinations that are out of their range. The Ad-hoc On Demand Distance Vector Routing (AODV) protocol is an algorithm used for the implementation of such networks. In this paper we discuss in detail about the functioning of AODV and how well it adapts to dynamic link conditions. More specifically, we compare the two implementations of AODV by Uppsala University (UU) and University of California, Santa Barbara (UCSB) and investigate the affect of node mobility on the performance of AODV. We show that the throughput graph achieved by the protocol is quite similar to the ideal throughput graph, in which the throughput increases as the packet size increases and saturates after a particular value of packet size

1. Introduction

An ad-hoc network is a collection of self-organized wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration stations unlike cellular wireless networks. The surrounding physical environment significantly attenuates and distorts the radio transmissions since signal quality degrades with distance. The effective transmission area of the node is limited and thus the effective throughput may be less than the radio's maximum transmission

capacity. Thus it makes it necessary for one mobile node to take the assistance of other nodes in forwarding its packets to the desired destination.

Mobile ad hoc networking supports multi-hop communication through IP routing. The initial approach for routing in MANETs were *proactive* i.e. the protocol constantly keeps a track of routes in the network and this requires the protocol to exchange control messages at a regular time interval. However, in MANETs, channel bandwidth and node energy are two important constrain factors and hence it is a good idea to use *reactive* routing, where routing is performed only on demand. The *Ad-hoc On Demand Distance Vector Routing (AODV)* protocol is one such algorithm. This paper discusses in detail the functioning of AODV and how well it adapts to the dynamic link conditions. More specifically, we compare the two implementations of AODV by Uppsala University (UU) and University of California, Santa Barbara (UCSB) and also the affect of node mobility on the performance of AODV. The main contribution of this paper is to provide real experimental results.

The rest of this paper is organized as follows. Section 2 covers an overview of a MANET routing protocols by explaining a proactive protocol, *DSDV (Destination Sequence Distance Vector)*, and a protocol, *DSR (Dynamic Source Routing)*. It is followed by a detailed description of AODV. Section 3 describes our experimental setup and the two scenarios (viz. *Static Scenario* and *Mobile Scenario*) we generated to evaluate the performance of AODV. In Section 4 we present the results obtained from our experiment. Section 5 concludes this paper and discusses our future work.

2. Related Work

In this section we briefly describe a proactive protocol, *DSDV (Destination Sequence Distance Vector)* [5,7], and two reactive protocols, *DSR (Dynamic Source Routing)* [6] and *AODV (Ad-hoc On Demand Distance Vector Routing)* protocol [1,5].

A proactive protocol exchanges topology information with other nodes of the network regularly. A particular node in the network announces the nodes, which are reachable by it periodically in their control messages so as to build and update the topology. On the other hand a reactive protocol initiates to find a route to the destination when needed. As compared to a proactive protocol in which routes are ready when

needed, the reactive protocol searches for routes only when needed. A network in which bandwidth is not a major constraint proactive protocols would be preferred since the lead time to start a transmission is less as routes to a destination are available instantly. However, in a MANET, the channel bandwidth is a major concern and hence using a proactive protocol would lead to a lot of routing overhead.

2.1 DSDV (Destination Sequence Distance Vector)

DSDV [5,7] is a proactive protocol and is based on the distance vector algorithm. Due to the dynamic topology of the network the nodes periodically broadcast routing updates. The routing table at each node keeps routing information about all the available destinations with the number of hops to that particular destination. To provide loop freedom DSDV uses sequence numbers, which is provided, by the destination itself. When a route to the next hop is broken the node immediately updates the sequence number and broadcasts the information to its neighbors. When a mobile node receives new routing information then it checks if it has a similar kind of information in its routing table. If the node already has that routing information then it compares the sequence number of the received information and the one it has. If the sequence number of the information it has is less than that of the received information then it discards the information with the least sequence number. If the both the sequence numbers are the same then the node keeps the information that has the shortest route or the least number of hops to that destination.

2.2 DSR (Dynamic Source Routing)

DSR [6] is a reactive protocol and uses the concept of source routing, which means that the source determines the complete path to the destination that the packets have to traverse, and hence ensures routing to be trivially loop-free. The packet in DSR carries route in its header thus permitting the intermediate nodes to cache the routing information in their route tables for their future use. The DSR discovers routes and maintains information regarding them by using two main mechanisms: *Route discovery* and *Route maintenance*.

Route discovery is the process that a source desiring to send data to a destination obtains a route to the destination if it does not have a route to the destination, and Route maintenance is the mechanism the node keeps track of the network topology i.e. it checks if any link breakage to a particular node has occurred or not. If a link breakage has occurred then the node tries to find another route to the destination or invokes Route Discovery for the same.

2.3 AODV (Ad-hoc On-Demand Distance Vector)

AODV [1,5] is a combination of DSR as well as DSDV. It uses the concept of route discovery and route mechanisms from DSR and uses the concept of sequence numbers, hop-by-hop routing and periodic beacons (i.e. hello messages) from DSDV. AODV is an on-demand routing protocol i.e. routes to the destination are only discovered when required thus avoiding memory overhead and less power. Moreover a node using AODV does not have to discover and maintain a route to another node until the two nodes need to communicate with once another. AODV uses destination sequence number, which is generated, by the destination itself for each route entry. The destination sequence number ensures loop freedom and if two similar routes to a destination exist then the node chooses the one with the highest sequence number. AODV uses *Route Request (RREQ)*, *Route Reply (RREP)*, and *Route Error (REER)* messages for route discovery and maintenance. The functioning of AODV is explained in the following subsections.

Generating and Handling RREQ

When a source wants to send information to a destination and does not have a route to it, then it generates a RREQ packet and broadcasts the packet to its neighbors. The RREQ uses the following fields in its packet: *Hop Count*, *RREQ ID*, *Destination IP Address*, *Destination Sequence Number*, *Originator IP Address*, and *Originator Sequence Number*. The hop count is the number of hops from the source to the node handling the RREQ. Thus when node receives a RREQ, if it is not the destination and nor does it have path to the destination it increments the hop count by 1 and rebroadcasts the packet to its neighbors. The destination IP address and the Originator IP address are the addresses of the destination and source generating the RREQ respectively. RREQ ID is a number that

uniquely identifies the RREQ. If the RREQ ID in the RREQ packet matches the RREQ ID in the nodes route entry table the RREQ will be dropped. Destination sequence number is the greatest sequence number received in the past by the originator for any route towards the destination.

When node receives the RREQ packet it checks to see if it is a destination, if it is not the destination, the node checks its routing table to see if it has a route to the destination, if it does, it (node) checks the destination sequence number in the RREQ packet and the one it has. If the destination sequence number it has is greater than the one in the RREQ then the node sends a RREP to the source stating that it has a route to the destination since a route associated with a higher Destination sequence number is regarded to be a fresher route to the destination. If the node does not have a route to the destination or if the node has a route but the sequence number associated with the route is less than that in the RREQ the node updates its routing table, increments the hop count by one to account for the new hop through this intermediate node, then creates a reverse route to the source by recording the address of the first neighbor from which it received the first copy of the RREQ and rebroadcasts the packet to its neighbors. When the destination receives the RREQ packet it prepares a RREP packet increments its current destination sequence number by one and sends the RREP packet to the source.

The source waits for the RREP for a fixed interval of time and then transmits the RREQ again and retries for a predefined number of times. If no response is received then the source declares that the destination is unreachable.

Route Table Management

The route table of a node maintains entries for each destination the node is interacting with or forwarding packets to. The routing table has the following fields: *Destination IP address, Active neighbors, Number hops, Next hop, Destination Sequence Number,* and *Expiration time for the routing table entry*. They help the node to maintain the connectivity of the network. The expiration time associated with the route depends on the size of the ad-hoc network and indicates the time after which the route to that associated destination in the route table is to be removed. The node maintains the list active neighbors that are the next hop to the destination associated in the route table, thus if a

link to this active neighbor is broken the node can immediately broadcast RERR messages.

Generating RERR Messages

A node broadcasts RERR packets when a link to the next hop is broken. This is how AODV reacts to link failures. Thus the node adds the destination addresses that are unreachable due to the link failure in the RERR packet and broadcasts it to its neighbors.
A RERR message is processed only when a node detects a link break for the next hop for which it has a an active route in its routing table or when it receives a RERR from the neighbor for an active route it has in its routing table or when it gets a DATA packet for a destination and it does not have an active route to the destination. Under these circumstances the node sends out RERR messages to its neighbors

Hello Messages

A node broadcasts Hello Messages periodically at a default rate of one per second, to maintain connectivity. These messages contain the nodes identity and sequence number to its neighbors so that its neighbors can update their local connectivity to the node that broadcasted the Hello Message. It can assume the link is broken and can broadcast a REER packet to its neighbors regarding the link failure. Other methods to maintain link connectivity are used, like physical and link layer methods to detect link breakages to nodes that it considers neighbors

3. Experimental Setup

The main focus of this paper is to practically implement AODV on a real system and evaluate the performance of the protocol. The AODV protocol was implemented on three Toshiba laptops (with Intel Pentium III processors) running the Red Hat Linux operating systems (version 7.1). We compared the performance of the two AODV implementations by UCSB [3] and UU [2]. We developed two scenarios, *Static* and *Mobile scenarios*, to evaluate the performance of AODV.

- Static scenario: As in Figure 1, Laptop 1 was kept at Position A while Laptop 2 and Laptop 3 were positioned at Position B and Position C respectively such that the end-to-end nodes were out of range of each other. The results obtained from this setup were used to plot the *throughput graph*, *network signature graph* and the *saturation graph*. All the laptops in this scenario were static and the experiment was conducted for **varying packet sizes.**

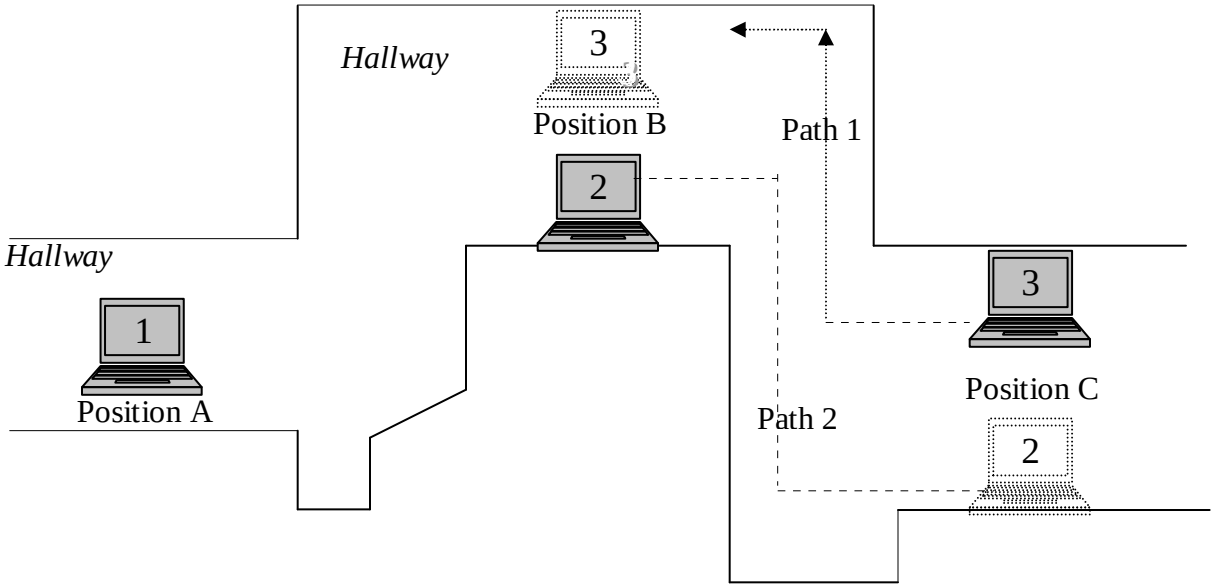

Figure 1: Experimental Scenarios: Static and Mobile

- Mobile scenario: For testing the affect of mobility on AODV, Laptop 3 was made to traverse Path 1 (as shown in Figure 1) from Position C to Position B along the hallway and simultaneously Laptop 2 was made to traverse Path 2 while Laptop 1 was kept at its original position. After Laptop 3 (2) reaches position B (C) (dashed font) it stays there for some amount of time after which it was taken back to its original position. When Laptop 3 is at Position B, it was in direct contact with Laptop 1. Laptops 2 and 3 were mobile in this scenario and the experiment was conducted for a **fixed packet size**.

The protocol was evaluated using software called *NETPIPE* [4], which provided a various parameters, using which the *throughput graph*, the *network signature graph*, *saturation graph* and the *mobility graph* were obtained as seen in the next section. The NETPIPE [4] software sends packets with increasing size and provides us with the following information: *Time taken to transfer the block*, *Throughput in bits/sec*, *Number of bits in the block transferred*, and *Number of bytes in the block transferred*. The idea behind increasing the packet size is to measure and compare the throughput for various packet sizes. Ideally for small packet size the throughput is less and with increasing packet size it increases until a point after which is saturates.

For the static scenario the experiment was conducted for various values of packet sizes and the Throughput Graph, Network Signature Graph and Saturation Graph were plotted using the above information provided by the software. For the mobile scenario the source code of the software was modified so that the packets size does not increase and hence the experiment was conducted for fixed packet sizes.

4. Results and Discussions

We implemented the AODV protocol designed by University of California Santa Barbara (UCSB) [3] and that designed by Uppsala University (UU) [2] and compared the throughput and latency for each implementation.

4.1 Static scenario

In this case the Laptop 1, 2 and 3 were static and were positioned at Position A, B and C respectively as seen from Fig.1. Laptop 3 was made to transmit packets to Laptop 1 via Laptop 2. Thus in all the three graphs to follow in this section Laptop 2 is the intermediate node.

Throughput graph

The Throughput Graph is a plot of **Throughput** vs. **Block Size** and is used to observe the maximum obtainable throughput of the protocol. As seen from Fig. 2 the throughput for AODV-UCSB gradually increases and stabilizes after a certain value of the block size while that of AODV-UU starts gradually but there is a steep rise as compared to that of

AODV-UCSB. It then increases gradually and then peaks for a few values of block Size as seen. AODV-UCSB stabilizes around 0.2 Mbps while AODV-UU stabilizes around 0.5 Mbps. The curves look similar too i.e. it gradually increases and then stabilizes which is quite similar to the ideal case. Small packets offer low throughput and as the packet size increases the throughput increases till a particular value, after which it saturates. In general both AODV-UCSB and AODV-UU give similar shaped throughput graph with AODV-UU performing better while AODV-UCSB having a shape quiet similar to that of an ideal case.

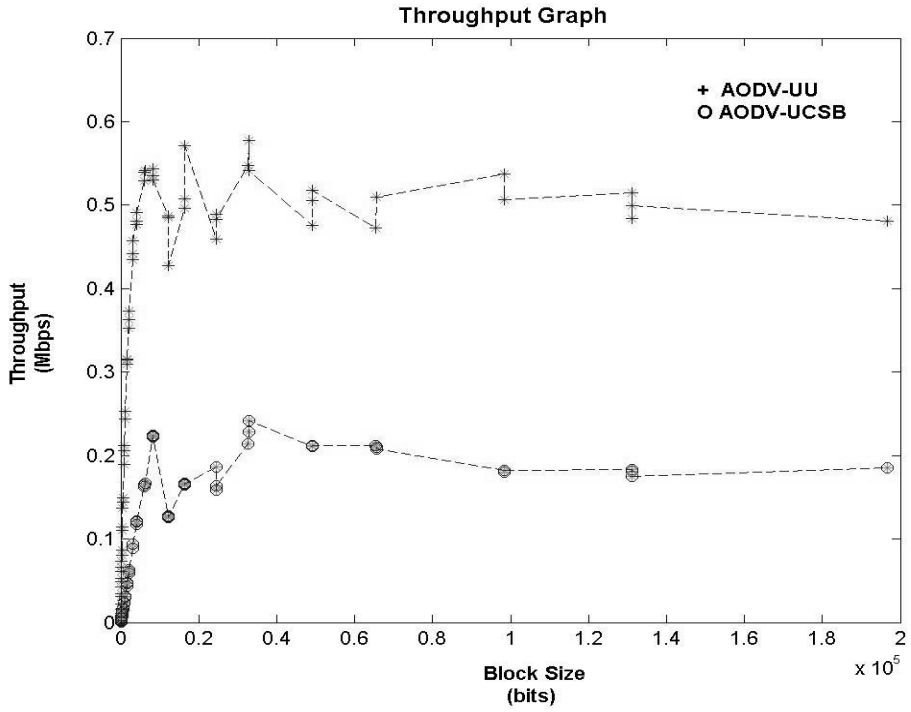

Figure 2: Throughput Graph comparison of UCSB to that of UU for Static scenario

Network Signature Graph

The graph below shows the network signature graph, which is a plot of **Throughput** vs. **Response Time,** used to observe the latency of AODV-UCSB and AODV-UU. The scale on the X-axis of the graph is Logarithmic while the one on the Y-axis is a normal scale. The scale on X-axis was changed to Logarithmic so that the latency between AODV-

UCSB and AODV-UU can be seen distinctly. As seen in the above graph the latency associated with AODV-UCSB is greater than that of AODV-UU. The latency associated with AODV-UU was found to be 0.002938 seconds while that in AODV-UCSB was 0.03200 seconds. Thus AODV-UU performs better since the latency associated with it is less than that of AODV-UCSB. The Point **A** on the graph indicates the response time (and corresponding Throughput) of UU to send the packet with size 196584 bits while the Point **B** on the graph indicates the response time (and corresponding Throughput) of UCSB to send a packet of size 196584 bits.

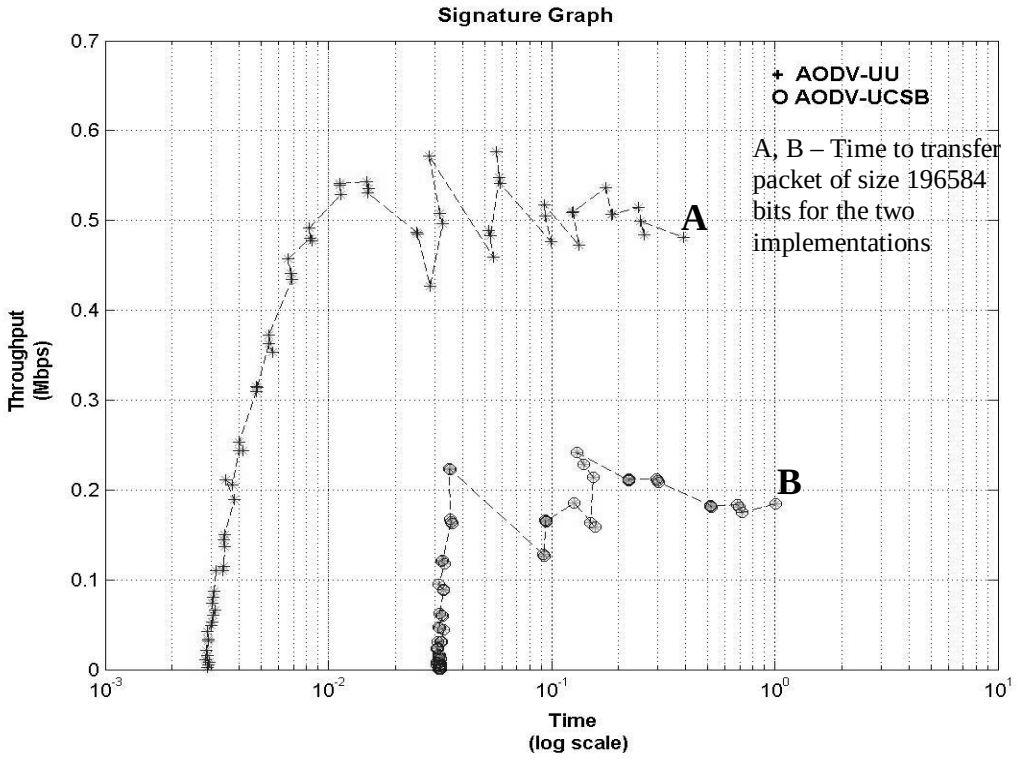

Figure 3: Network Signature Graph Comparison for Static scenario

Saturation Graph

The Saturation graphs are a plot of **Block size** and **Response time** to transfer a block and are used to analyze saturation point after which there is a linear increase. As seen in Fig.4 above as the block size increases the increase in time is gradual. After a certain value of block size the time increases linearly with the increase in block size, which is effectively

the knee of the curve. The point after which this happens is called the *Saturation Point* while the interval between the saturation point and the end of the recorded data is referred to as the *Saturation interval*. As seen in the graph the saturation point is circled and the Saturation interval is the straight line indicating the linear increase between the block size and the time. Thus during the saturation interval the graph increases monotonically at a constant rate and this indicates that the Throughput cannot be improved upon increasing the Block Size.

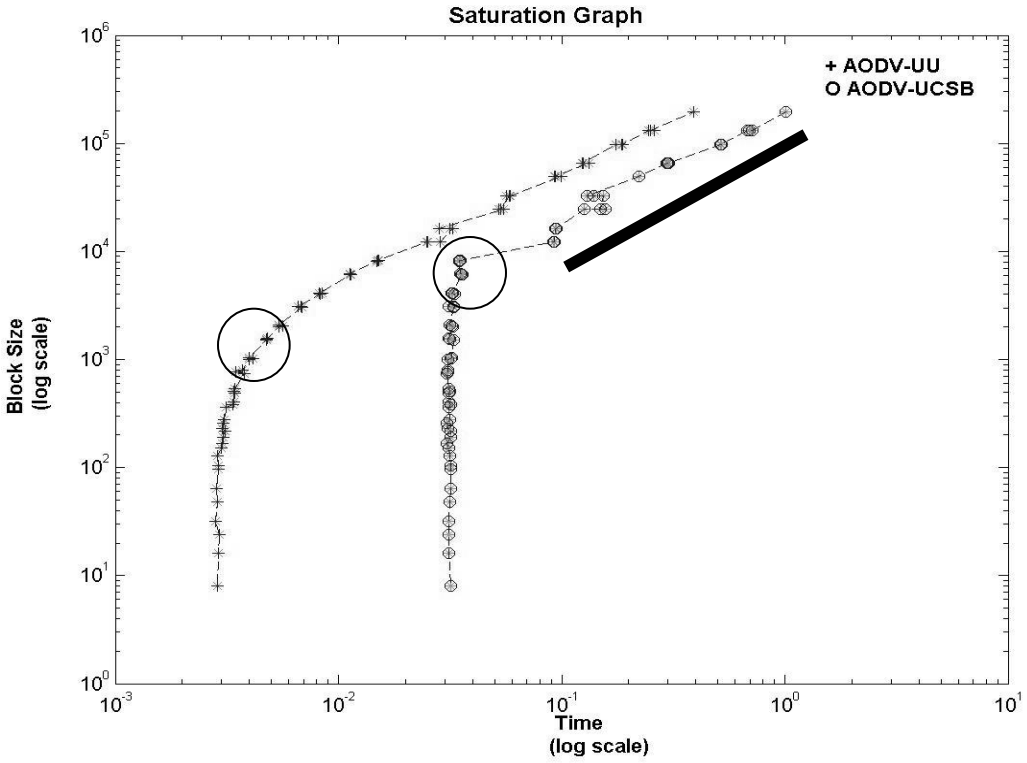

Figure 4: Saturation Graph Comparison of UU to that of UCSB for Static scenario

4.2 Mobile Scenario

To obtain the affect of mobility on the performance of AODV we conducted the experiment with a fixed packet size of 1024 bytes. The packet was transmitted 70 times and the observations were noted. The speed provided to the Laptops was the walking

speed of a normal person. The Laptop 3 was initially positioned at Position C as seen in Fig.1. The experiment was started and Laptop 3 was made to transmit packets to Laptop 1 via Laptop 2. After 12 packets were transmitted Laptop 1 was made to traverse Path 1 and brought to Position B as seen in the Fig.1 by the dashed font and simultaneously Laptop 2 was made to traverse the Path 2 as shown in dashed font in Fig.1 to Position C. Thus the link between Laptop 1 and Laptop 3 was broken since Laptop 2 was not in range of Laptop 1. Thus Laptop 3 had to find a new route to Laptop 1, since Laptop 1 is now in direct range of Laptop 3 a link between the two is formed and the data is transmitted.

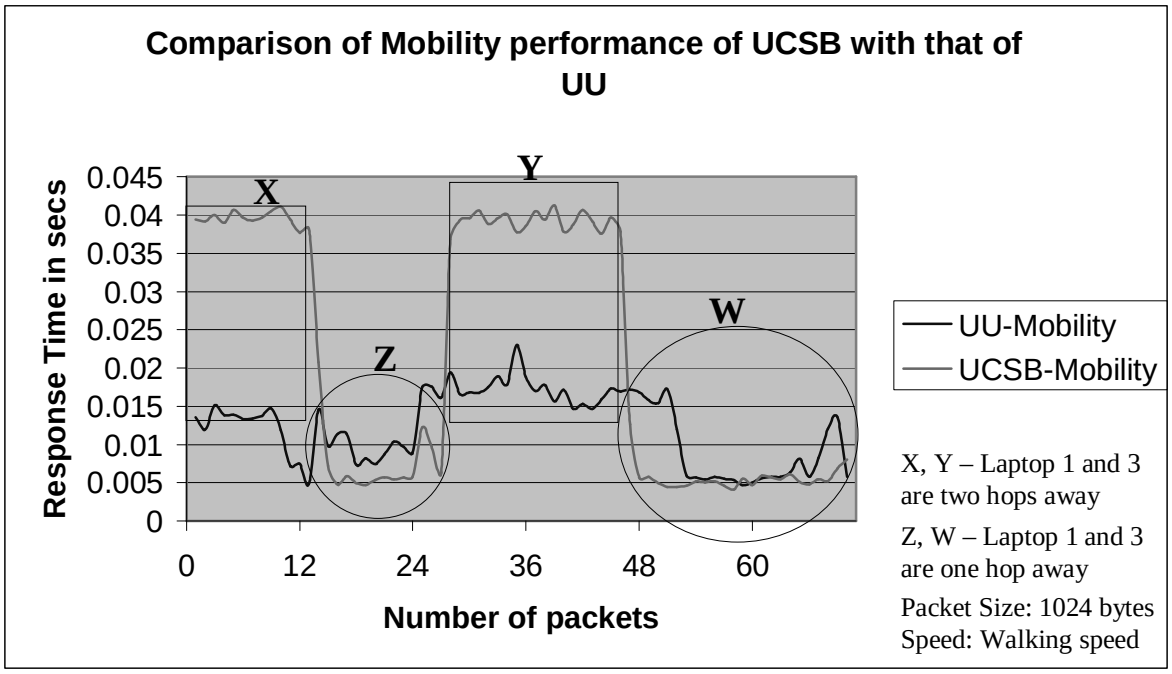

Figure 5 Node Mobility Graph for Mobile Scenario with a Fixed Packet Size

As seen from the Fig.5, time to send the packet decreases drastically since the two Laptops are in direct transmission range of each other. Circle 1 and Circle 2 represent the period when the two laptops are in direct transmission range of each other. The Dashed rectangles X and Y indicate the period for which the two laptops are one hop away from each other. After approximately 25 packets had been transmitted Laptop 3 and Laptop 2 were brought back to their original positions i.e. Position C and Position B respectively.

Thus as seen from the graph the (Dashed Rectangle Y) the time required transmitting the packet increases since, Laptop 1 and Laptop 3 are one hop away. After approximately 50 packets had been transmitted Laptop 3 and Laptop 2 were made to traverse Path 1 and Path 2 respectively as seen in Fig.1. Our aim is to see how fast AODV responds to link breakage, hence as seen in Fig.5 above both AODV-UU and AODV-UCSB respond to link breakage quite promptly since there is a steep fall and rise when the laptops switch positions.

5. Conclusions and Future Work

We implemented two implementations of the AODV protocol viz. UU and UCSB on real systems and compared the two implementations based on the throughput and latency. As seen from Fig. 2 the throughput achieved by AODV-UU is higher than that of AODV-UCSB and from Fig. 3 we can conclude that latency associated with AODV-UCSB is greater than AODV-UU. Thus AODV-UU seems to perform better than AODV-UCSB. To evaluate the performance of AODV protocol to link breakages we provided mobility to the Laptops so as to force a link breakage during packet transmission so that the node has to find another route to the destination. Thus as seen from Fig. 5 AODV responds to link breakage quite promptly due to the steep rises and falls of the graph. Well both AODV-UU and AODV-UCSB respond equally well to link breakage and promptly find another path to their destination.

In ad hoc network the topology changes very frequently and moreover the nodes are on a constant move. Thus our future works consists of measuring the performance of AODV based on the mobility of the nodes and with respect to the pause time and also modifying NETPIPE so that we can get the exact time required for a new link to be established. Ideally an ad hoc network could consist of 50-100 nodes or more, but implementing such a large network practically is out of our scope due to limited hardware and labor. Thus we propose to form a small ad hoc network of about 6-8 nodes and provide mobility to them so that the network topology keeps changing. We can then measure the throughput, latency, packet loss rate, routing overhead etc. associated with AODV. We will also try to simulate the Ad-hoc network in simulation and compare the difference in results obtained.

References

[1] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir Das. "Ad Hoc On Demand Distance Vector (AODV) Routing." *IETF Internet draft*, draft-ietf-manet-aodv-10.txt, March 2002 (Work in Progress).

[2] Implementation of AODV, Uppsala Univerity http://user.it.uu.se/~henrikl/aodv/

[3] Implantation of AODV, University of California, Santa Barbara http://moment.cs.ucsb.edu/AODV/aodv.html

[4] NETPIPE, http://www.scl.ameslab.gov/Projects/Netpipe/

[5] Charles E. Perkins and Elizabeth M. Royer. "Ad hoc On-Demand Distance Vector Routing." *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90-100.

[6] Dynamic Source routing protocol Internet draft. http://www.ietf.org/html.charters/manet-charter.html

[7] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994. A revised version of the paper is available from http://www.cs.umd.edu/projects/mcml/papers/Sigcomm94.ps