

Performance Analysis and Comparison of Ad-Hoc Routing Protocols

Ismail Ari , Neelu Jethani, Aniruddha Rangnekar, Sushma Natarajan
{ari, njetha1, arangn1, snatar3}@mctr.umbc.edu

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County

CMSC 691T, MOBILE COMPUTING, PROJECT REPORT

May, 22, 2000

Abstract

As of date, wireless communication is one of the most demanding areas of research within networking, with hundreds of ill-defined; yet unanswered questions, proposed, but unverified protocols and with great expectations from the industry and users.

The success of the proposed protocols depends on the availability of robust implementations that enable both real-time testbeds and non-real-time simulations. The purpose of this research is to fill this gap by comparing the protocols recently proposed routing for wireless communications environment, with freely moving mobile hosts forming groups of various sizes and called an *ad-hoc network*.

1 Introduction

Ad-hoc networks are self-organizing, multi-hop wireless networks. Ad-hoc networks are rapidly deployable, since they do not have a fixed infrastructure. All mobile hosts in an ad-hoc network are embedded with packet forwarding capabilities. Typical applications of ad-hoc networks are outdoor special events such as conferences, concerts and festivals; places with no network infra-structure; cases of emergencies and natural disasters; and military maneuvers. The characterizing features of ad-hoc networks are the highly dynamic topology and the very limited resources of bandwidth and computational power. These unique features pose several new challenges in the design of wireless, ad-hoc networking protocols.

Our goal is to study the different routing protocol proposals for ad-hoc networks. Namely, Dynamic Source Routing protocol (DSR), Ad-hoc On-Demand Distance Vector Protocol (AODV), Destination-Sequenced Distance Vector (DSDV) and Core Extraction based Distributed Ad-hoc Routing (CEDAR). We used ns2¹ network simulator with wireless extensions from Carnegie Mellon University for this purpose.

Section 2 presents the ad-hoc routing protocols that will be compared. Section 3 describes our simulation methodology and the metrics of interest for the performance analysis. Section 4 presents the simulation results and compares the protocols both quantitatively and qualitatively. Section 5 focuses on the CEDAR protocol performance and Section 6 draws the conclusion explaining which protocol is more suitable for various environments.

2 Background

This section describes some of the most prominent the ad-hoc routing protocols in the ad-hoc networking community today.

¹ <http://www-mash.cs.berkeley.edu>

2.1 DSDV (Destination Sequenced Distance Vector)

DSDV is a hop-counting distance vector protocol requiring each node to periodically broadcast routing updates. The main advantage of DSDV over traditional distance vector routing protocols is that it guarantees loop freedom.

In DSDV, each node maintains a routing table that has an entry for each destination in the network. The attributes for each destination are the next hop ID, hop count metric and a sequence number which is originated by the destination node. DSDV uses both periodic and triggered routing updates. Triggered routing updates are used in addition to the periodic updates in order to propagate the routing information as quickly as possible whenever there is any topological change. The update packets include the destinations accessible from each node and the number of hops required to reach each destination along with the sequence number associated with each route.

Upon receiving a route update packet, each node compares it to the existing information regarding the route. Routes with old sequence numbers are simply discarded. In case of routes with equal sequence numbers, the recently advertised route replaces the old one if it has a better hop count metric. The metric is then incremented by one, since the incoming packet will require one more hop to reach the destination. Newly recorded routes are immediately advertised to the neighbors.

When a link to a next hop is broken, any route through that next hop is immediately assigned an infinite value with a new sequence number. When a node receives an infinite count and has an equal or greater sequence number with a finite metric, a route update broadcast is triggered. Thus, real routes propagated from the newly located destination will quickly replace the routes with infinite value.

DSDV also employs a mechanism to damp out fluctuations in route table updates. In an environment where many independent nodes transmit routing information asynchronously fluctuations could arise. For example, a node could receive two routes to the same destination with the same sequence number, but say the one with the worse metric always arrives first. This could lead to continuous outbursts of route updates. DSDV solves the problem by using "*settling time*" data. Specifically, time duration until the route becomes *stable* (termed settling time) is predicted, and the settling time is allowed before advertising any new route information to the network. In other words, the settling time is used to decide how long to wait before advertising new routes. Delaying the advertisement of unstable routes, fluctuations in the routing tables are prevented and thus the number of route updates is reduced.

The protocol has a number of drawbacks. Optimal values for the parameters like maximum settling time for a particular destination are difficult to determine. This might lead to route fluctuations and spurious advertisements resulting in waste of bandwidth. DSDV uses both periodic and triggered routing updates, which could also cause excessive communication overhead. In addition, in DSDV a node has to wait until it receives the next route update originated by the destination before it can update its routing table entry for that destination. Finally, DSDV does not support multi-path routing.

2.2 DSR (Dynamic Source Routing)

DSR is based on source routing, where the source specifies the complete path to the destination in the packet header and each node along this path simply forwards the packet to the next hop indicated in the path. It utilizes a route cache where source routes it has learned so far are cached. Therefore, a source first checks its route cache to determine the route to the destination. If a route is found, the source uses this route. Otherwise, the source uses a route discovery protocol to discover a route. A route is sought only by a source when needed.

In route discovery, the source floods a query packet through the ad-hoc network, and the reply is returned by either the destination or another host that can complete the query from its route cache. Each query packet has

a unique ID and an initially empty list. Upon reception of a query packet, if a node has already seen this ID (i.e. it is a duplicate) or if it finds its own address already recorded in the list, it discards the copy and stops flooding; otherwise, it appends its own address to the list and broadcasts the query to its neighbors. If a node can complete the query from its route cache, it may send a reply packet to the source without propagating the query packet further. Furthermore, any node participating in route discovery can learn routes from passing data packets and gather this routing information into its route cache. This route discovery protocol is similar to the Internet's Address Resolution Protocol (ARP) except that ARP requests do not propagate beyond a router.

A route failure can occur since mobile hosts move from one place to another. A route failure can be detected by the link-level protocol (i.e. hop-by-hop acknowledgments), or it may be inferred when no broadcasts have been received for a while from a former neighbor. When a route failure is detected the node detecting the failure sends an error packet to the source, which then uses the route discovery protocol to find a new route. Note that in DSR, no periodic control messages are used for route maintenance.

The major advantage of DSR is that there is little or no routing overhead when a single or few sources communicate with infrequently accessed destinations. In such situation, it does not make sense to maintain routes from all sources to such destinations. In DSR, only the sources that desire communication with such destinations need to discover those routes. Furthermore, since communication is assumed to be infrequent, a lot of topological changes may occur without triggering new route discoveries (i.e. has little or no communication overhead).

Even though DSR is suitable for the environment where only a few sources communicate with infrequently accessed destinations, it may result in large delays and large communication overheads in highly dynamic environments. Therefore, DSR may have dynamic scalability problem. As the network becomes larger, control packets and message packets also become larger, since they need to carry the addresses of every node in the path. This may be a problem, since ad-hoc networks have limited available bandwidth.

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

Route Requests (RREQs) and Route Replies (RREPs) are the two message types defined by AODV. When a route to a new destination is needed, the node uses a broadcast RREQ to find a route to the destination. A route can be determined when the request reaches either the destination itself or an intermediate node with a fresh route to the destination. The route is made available by unicasting a RREP back to the source of the RREQ. Since each node receiving the request keeps track of a route back to the source of the request, the RREP Reply can be unicast back from the destination to the source, or from any intermediate node that is able to satisfy the request back to the source.

A hello message is a local advertisement for the continued presence of the node. Neighbors that are using routes through the broadcasting node will continue to mark the routes as valid. If hello messages from a particular node stop coming, the neighbor can assume that the node has moved away. When that happens, the neighbor will mark the link to the node as broken and may trigger a notification to some of its neighbors telling that the link is broken.

In AODV, each router maintains route table entries with the destination IP address, destination sequence number, hop count, next hop ID and lifetime. This information must be kept even for ephemeral routes, such as those created to temporarily keep track of reverse paths towards nodes originating the RREQs.

2.4 CEDAR (Core Extraction Distributed Ad-hoc Routing)

CEDAR is an algorithm for QoS routing in ad-hoc network environment. CEDAR dynamically establishes a core for the network and incrementally propagates link state of stable high bandwidth links to the nodes of the core. Route computation is on-demand and is performed by core hosts using local states only. CEDAR is proposed as a QoS routing algorithm for small to medium size ad-hoc networks consisting of tens to

hundreds of nodes. CEDAR is a robust and adaptive QoS routing algorithm that reacts quickly and effectively to the dynamics of the network, while still approximating the link state performance of stable networks. The following is a brief description of the three key components of CEDAR:

1.Core Extraction: CEDAR does core extraction in order to extract a subset of nodes in the network as the only ones to perform state management and route computation. The core extraction is done dynamically by approximating a minimum dominating set of the ad-hoc network using only local computation and local state. Each host in the core then establishes a virtual link (via a tunnel) to nearby core hosts (within the 3rd neighborhood in the ad-hoc network). Each core host maintains the local topology of the hosts in its domain, and also performs route computation on behalf of these hosts. The core computation and core management upon change in the network topology are purely local computations that enable the core to adapt efficiently to the dynamics of the network.

2.Link State Propagation: While it is possible to execute ad-hoc routing algorithms using only local topology information at the core nodes, QoS routing in CEDAR is achieved by also propagating the bandwidth availability information of stable links in the core. The basic idea is that the information about stable high-bandwidth links can be made known to nodes far away in the network, while information about dynamic links or low bandwidth links should remain local. By means of this link state propagation mechanism, CEDAR approximates a minimalist local state algorithm in highly dynamic networks while it approaches the link state algorithm in highly stable networks. The propagation of link-state is achieved through slow-moving 'increase' waves (which denote increase of bandwidth) and fast-moving 'decrease' waves (which denote decrease of bandwidth). The key questions to answer in link state propagation are: when should an increase/decrease wave be initiated, how far should a wave propagate, and how fast should a wave propagate.

3.Route Computation: Route computation first establishes a core path from the domain of the source to the domain of the destination. This initial phase involves probing the core and caching the resultant core path for future use. The core path provides the directionality of the route from the source to the destination. Using this directional information, CEDAR iteratively tries to find a partial route from the source to the domain of the furthest possible node in the core path (which then becomes the source for the next iteration) that can satisfy the requested bandwidth. Effectively, the computed route is a concatenation of the shortest-widest (the least hop path among the maximum bandwidth paths) paths found locally using the core path as the guideline.

3 Methodology

We used the discrete event NS-2 network simulator for analysis and comparison of the ad-hoc routing protocols. The mobility model uses the *random waypoint model* in a rectangular field.

Random waypoint model has been the choice in almost every prior ad-hoc routing protocol analysis. We will try to discuss how well it represents real life cases in Section 6 by simulating a real life scenario. We will try to demonstrate or refute the validity of random waypoint model assumption.

We used various sized areas for mobile node movements. Some examples are 500mx500m , 1500mx500m, 1500mx300m etc. Each node starts its journey from a random location and moves to a random destination with a randomly selected speed that is uniformly distributed between 0-20m/sec. Once the destination is reached another random destination is targeted after a pause. We have used pause times ranging from 0 seconds (continuous mobility) to 900 seconds in increments of 100 seconds during the 900 second simulation time. For the 900 seconds pause case all nodes would remain stationary during the simulation.

The sources are CBR (Continuous Bit Rate) traffic generators. The data packet sizes are 512 bytes. The number of traffic generators (called "*connections*" in other papers) and the packet generation rates are varied to test various load conditions in the network. Three key performance metrics are:

- **Throughput** : Ratio of the packets delivered to the total number of packets sent.
- **Average End-to-End Delay** : Time taken for the packets to reach the destination
- **Routing Overhead**: The number of control packets for the routing protocol over the number of data packets sent.

4 Results and Discussion

In this section we will analyze and compare the protocols based on four main parameters: the effects of changing pausetime (sec), speed of nodes (also called mobility in m/s), bit rates (KBps) and number of traffic generators (also called connections) on protocol throughput, protocol routing overhead and average end-to-end delay for data packet delivery.

4.1 Pausetime (sec)

Nodes will stop a “*pausetime*” amount before moving to another destination point. Note that we shift from very dynamic towards static topologies as the pausetime between movements increases from zero to the final total simulation time.

Fig. 1 shows the relative throughput performance of the three routing protocols as pausetime for 20 nodes changes. When nodes pause for larger times, the protocols deliver a greater percentage of the originated data packets. When there is no mobility all protocols achieve 100% throughput. DSR and AODV perform well in all cases, delivering an average of 98% of the data packets. However, DSDV throughput degrades to 80% as the pausetime gets smaller, since a stale routing table entry causes data packets to be forwarded over a broken link. DSDV maintains only one route per destination, so each packet that the MAC layer is unable to deliver is dropped due to the lack of alternate routes. These results are similar to the results found by the Monarch group in CMU.

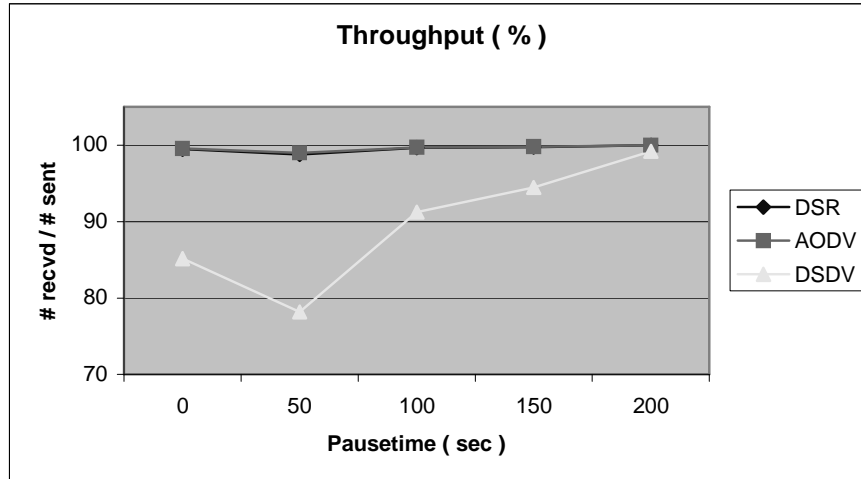


Figure 1: Throughput decreases significantly in DSDV as mobility increases (nodes pause less).

Fig. 2 shows the effect of increasing mobile node number on the throughput of the DSR protocol. When the number of nodes in the network is high, the topology is dense and the connectivity is rich. Hence, with larger number of nodes DSR performs well even with small pausetimes, i.e. higher mobility. Networks suffer from partitions with fewer nodes. There is no remedy for a partition. Packets will have to be dropped if no connection can be found. When nodes start dropping packets the throughput decreases down to 60%. What is not shown in these graphs is that, as the node number increases more routing decisions is made and the number of routing packets exchanged increases.

The protocols impose different amounts of routing overhead, as shown in Fig. 3. DSR has the least routing overhead among the three at all times and the routing overhead increases slightly as mobility increases.

DSDV imposes a constant overhead to the network at all pausetimes because of the periodic nature of the routing updates.

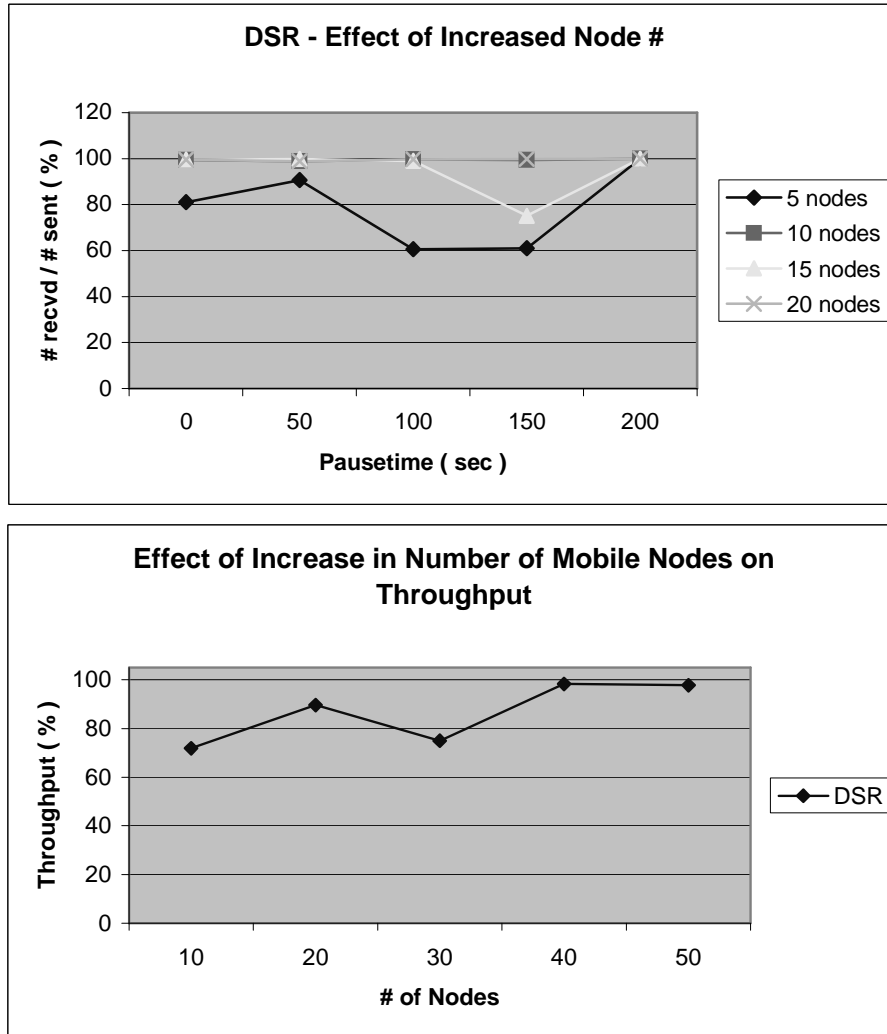


Figure 2: (Top) When connectivity between nodes decreases throughput also decreases. (Bottom) This graph shows the throughput change in DSR vs. node number for a single pausetime.



Figure 3: Protocols have different routing overhead characteristics.

We have seen in Fig. 2 that AODV throughput did not degrade with increase in pausetime. In Fig. 3 the hidden cost of keeping the throughput constant becomes apparent. The routing overhead of AODV almost doubles at each step of the decrease in pausetime. So, although AODV incurs comparable overhead to DSR protocol at low mobility, the overhead explodes when mobility is high.

4.2 Mobility (m/s)

Although implicitly related to the pausetime metric used in part A of this section, we found it relevant to use another terminology for the “mobility” of the nodes, which basically shows how fast the nodes are moving. We will consider a wide range of speeds for our mobile nodes from 1 m/s (3.6 km/hour), that corresponds to walking at a slow pace, to 50 m/s (180 km/hour), the speed of a very fast car.

We first look at the effect of increased speed on throughput. Fig. 4 shows that all of the protocols have higher throughput (DSR and AODV have 100% while DSDV has 96%) when the nodes move at low speeds. When the speed increases, all the routing protocols suffer a decrease in throughput. Higher speeds cause frequent link changes and connection failures. Overall performance of DSR and AODV is better than DSDV. DSDV drops about 25% of the packets when the speed is increased to 50m/s. Since DSDV maintains only one route per destination, packets that cannot be delivered by the MAC layer are dropped due to the lack of alternate routes.

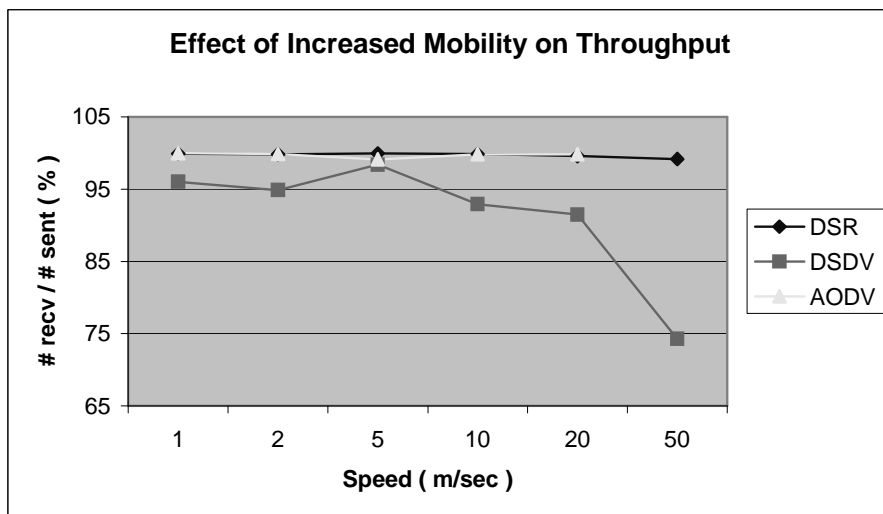


Figure 4 : DSDV cannot handle mobility at high speeds due to the lack of alternate routes when the route in routing table is stale.

Next, we measure the effect of increasing node speed on routing overhead. Fig.5 shows that the overall routing overhead to be 10-20% for all the routing protocols, which is actually close to ideal for desired in-band signaling ratios in any networking protocol. However, DSDV trades off this constant routing overhead with a decrease in throughput as shown in Fig. 4. Note that the routing overhead for DSDV remains constant as the speed increases. However, for DSR and AODV the routing overhead slightly increases.

In Fig. 6 it can be seen that increase in node speeds results in significant increase in the average end-to-end packet delivery delay of AODV protocol. This is because when a node receives a route request for which it has the answer in its routing table, it immediately replies with the route rather than forwarding it to the destination. The source can now start to communicate with the destination. Since AODV maintains only one routing entry per destination, it has to do more route discoveries as the speed increases. Therefore, the average delay increases as the time taken to find a route to the destination increases when there is no entry for it in the intermediate nodes.

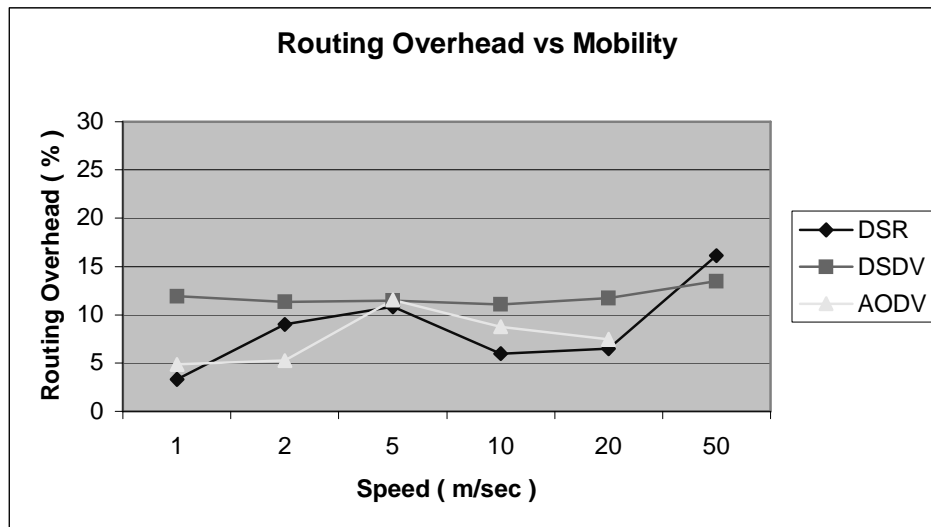


Figure 5 : Almost all protocol achieve an ideal 10 % stable routing overhead that is not affected by the speed of nodes.

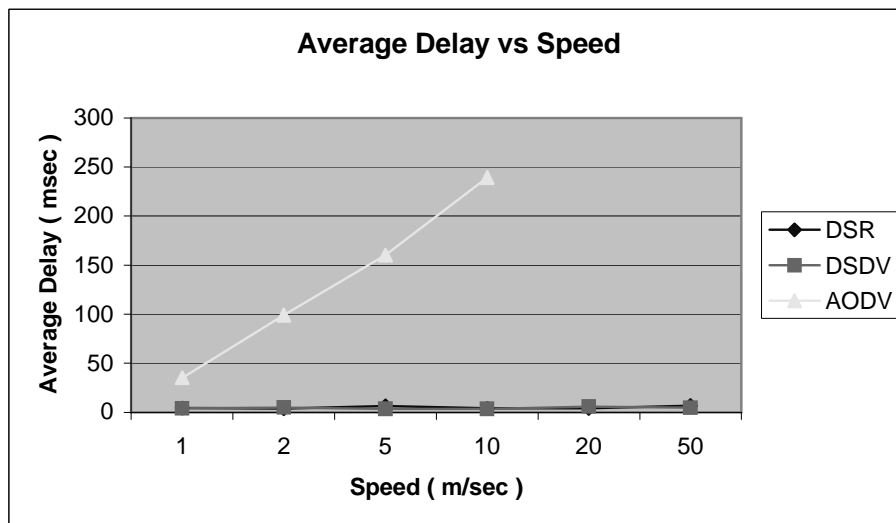


Figure 6 : AODV data packet delivery delay increases significantly with increased speed.

4.3 BIT-RATE (Application Message Generation Rate)

In this section we will try pumping as much data as we can into the network from the mobile nodes and try to find how protocols behave in terms of throughput, routing overhead and packet delivery delay. Using the terminology of *queuing theory*, we will increase the *arrival rate* and observe the limits for *service rate*.

First, note that the throughput decreased in all protocols in Fig. 7 with increased bit-rate. The degradation is largest in AODV. The reasons are similar to those discussed for AODV above.

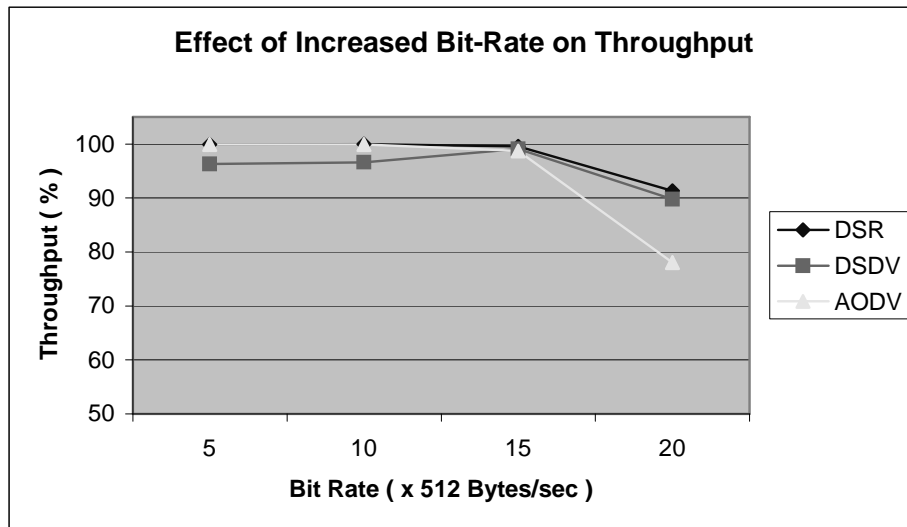


Figure 7 : AODV throughput is effected the most from bit-rate increase

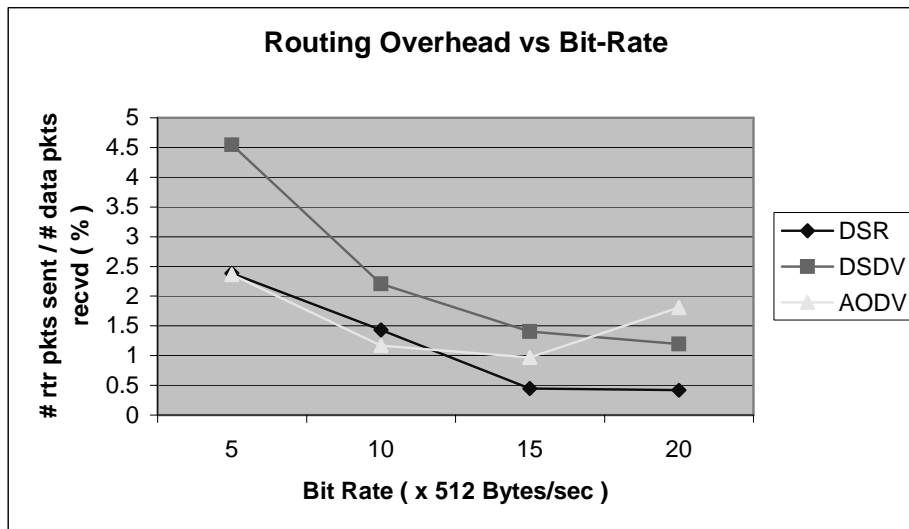


Figure 8 : More packets are delivered with one route discovery. This decreases routing overhead ratio. AODV could best handle up to 7.5KB/sec with this configuration.

Intuitively, we would expect the routing overhead to increase as the bit-rate (message generation rate) increases. But as seen in Fig. 8 the routing overhead actually decreases as the bit-rate increases. This is because as the bit rate is increased, for every router packet sent the number of data packets that is received increases. Only one route discovery packet is used for finding out the route for more data packets and the ratio decreases. The routing overhead for AODV increases as the bit rate increases from 7.5KB/s to 10KB/s. This shows the limit of traffic AODV protocol can handle for this configuration.

Fig.9 shows that as the bit-rate increases the average end-to-end delay also increases, i.e. the time taken for the packets to reach the destination increases. The increase in delay for AODV is more than that in DSR and DSDV. More routing packets need to be sent for every data packet sent in AODV. Thus it takes more time to establish a route in AODV.

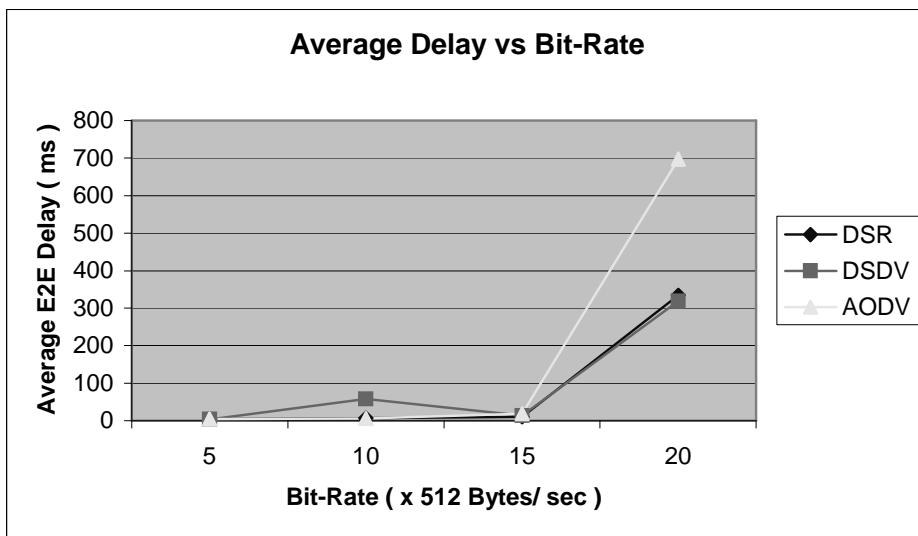
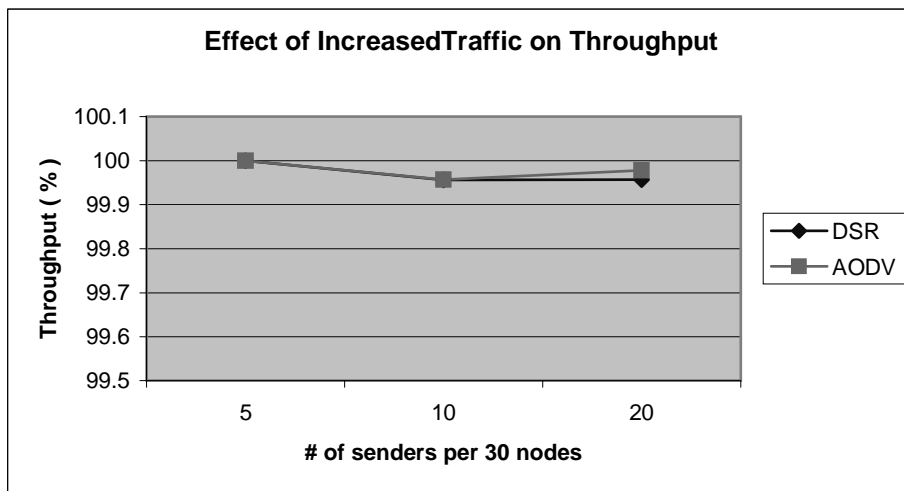


Figure 9 : Average end-to-end delay also increases as the bit-rate increases.

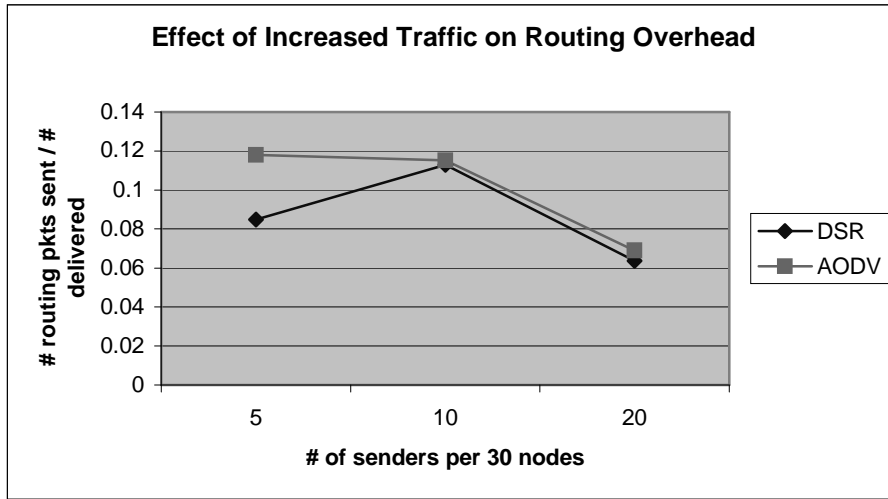
4.4 Number of Traffic Generators

The simulations in this part test some claims made by Charles Perkins in his AODV paper. He comments that DSR cannot handle large number of connections and the merits of AODV become apparent for these high traffic load cases. Our comparison shows that for 30 nodes DSR throughputwise performs as good as AODV (a) with less routing overhead (b) and less average end to end delay (c) at all times. However, our scenario is not exactly the same as the scenario used in that paper.

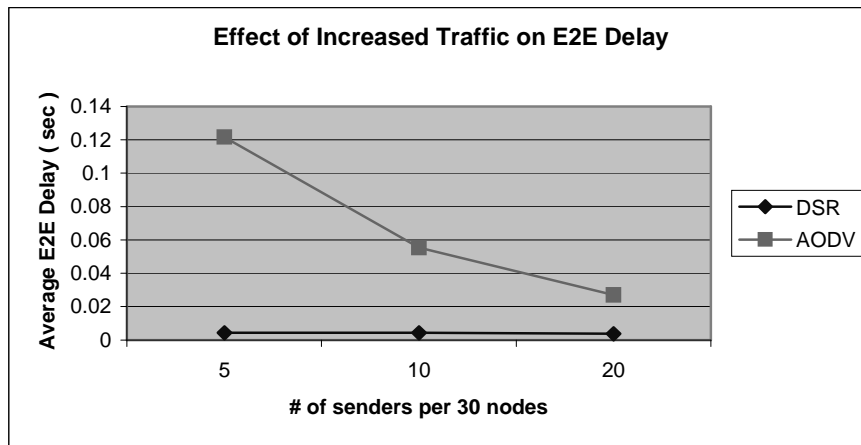
Figure 10: Effect of increased number of traffic generators on (a) throughput (b) routing overhead and (c) average end-to-end data packet delivery delay.



(a)



(b)



(c)

5 A Closer Look At CEDAR Protocol

We would like to thank and credit TIMELY group in UIUC for their valuable contributions for the values used in this section. CEDAR has been compared with AODV protocol in these two sets of graphs. On the left hand side mobile nodes are spanning an area of size 1500m x 300m and on the right hand side an area of size 1500m x 1500m.

When the area size is small (left) the throughput obtained by CEDAR is greater than AODV and still with less routing overhead, regardless of whether the periodic broadcast of beacon messages in the core have been turned ON or OFF. Of course, note that the addition of periodic beacons (part b) adds some more routing overhead to CEDAR compared to the case without beacons (part c).

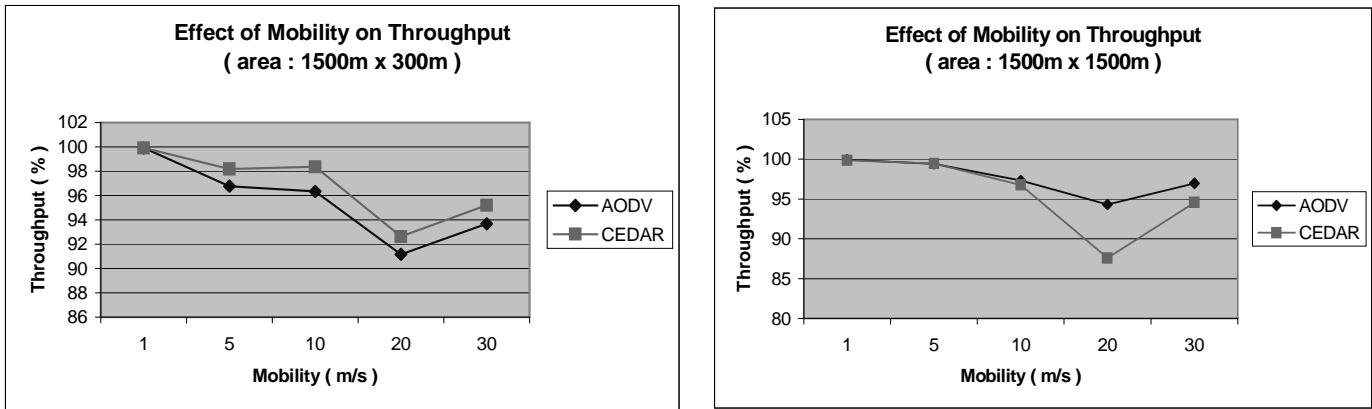
The more interesting result is that when the area gets larger CEDAR is performing worse than AODV both in throughput and routing overhead, as seen on right hand side. We believe this is due to the fact that the size of the spanned area increases as the network topology becomes sparser. Thus, more nodes are forced to become a member of the core since they cannot easily find themselves a nearby dominant node that is already member of the core itself.

One would think that CEDAR topologies (core + others) should converge to a flat network topology in sparse mode and perform just inferior as other ad-hoc routing protocols, but not worse. However, it turns out that allowing the communication of the leaf nodes only through the core nodes may block some possible

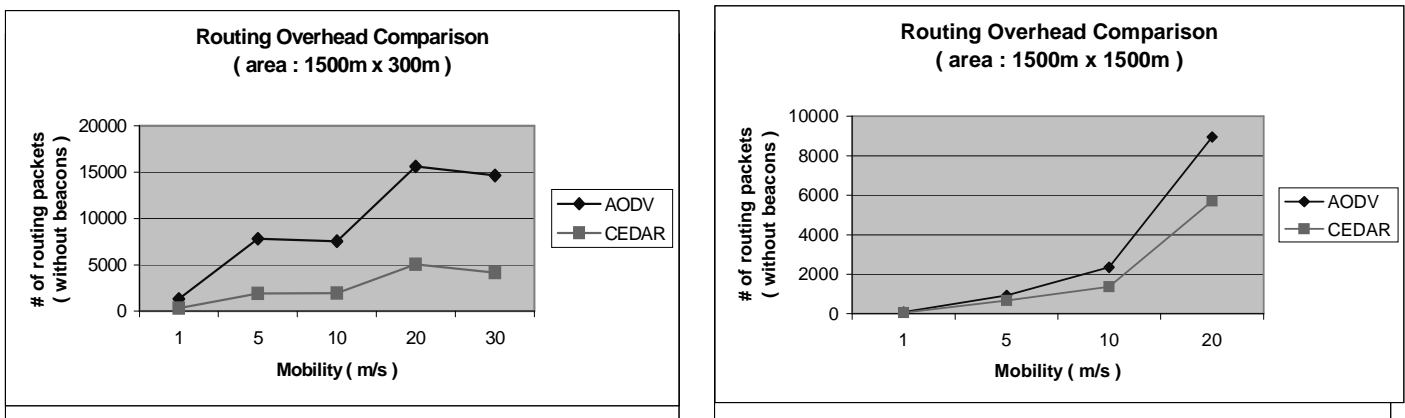
transmissions that were otherwise possible and this causes degradation in throughput. The idea here is that: leaving every node take care of itself is better than limiting its communication through some other node in very sparse cases.

The cost of maintaining a core (i.e. the routing overhead) also increases as number of nodes in the core increases, since more core nodes means more message exchange and sparser topology means establishment of more virtual links. These mechanisms need specialized packet exchanges and altogether add up to the total routing overhead. We argue that this outcome altogether contradicts with the notion of keeping a hierarchical structure, since hierarchy is usually needed to give support to larger sized networks.

(a) Throughput Comparisons



(b) Routing Overhead Comparison without beacons



(c) Routing Overhead Comparison with beacons

Figure 11 : Although core notion helps CEDAR to achieve higher throughputs with lower overhead in small areas (therefore dense topologies) , core performs worse than a flat network topology both in respect to throughput and overhead in large areas, where the topology is sparse.

6 A Realistic Ad-Hoc Routing Scenario (UMBC Case Study)

In this section we will try to investigate the validity of the random waypoint movement model that has been used by almost every ad-hoc routing protocol comparison effort up to date. To be able to do this we will generate a real life ad-hoc movement scenario, namely the movement of students between most popular places around University of Maryland Baltimore County (UMBC) Campus shown in Fig.12. The size of the area shown in the map is almost exactly the same we are using for our simulations.

6.1 Motivations

Random movement patterns are used commonly for simulation of ad-hoc networks. Random movement scenarios are believed to test “worst case scenarios”. However, in ad-hoc networks these movements usually provide a smooth mobile-node distribution and have the danger of hiding the problems that could be observed in real-life movement scenarios. Fig. 13 shows the snapshot of a simulation we ran for 50 nodes, 900 total simulation time, 40 sources and 4 packets/sec transmission rate (corresponding two 16Kbps since we are using 512Byte packets) in an area of 1500m by 300m (50-900-40-4.nam).

The dependency of throughput on the network topology and thus the movement scenario becomes much clear in the simulation results shown in Fig.14 for DSR. To begin with the network with 10 nodes is too sparse in an area of size 1500mx300m, so generally the network is prone to partitions and the throughput is very low. When the pausetime is 900sec nodes complete the simulation without any movement. With 800sec pausetime they move once and spend time in the second topology 100 seconds until the end. Apparently, in our simulation we have noticed that the first topology had some partitions and the movement to second position at 800 second helped resolve this pathological case. That is why the throughput increases from 900 second as we go towards 500 sec pausetime. The third movement also caused network to be partitioned and degraded the throughput for 300 sec pausetime.

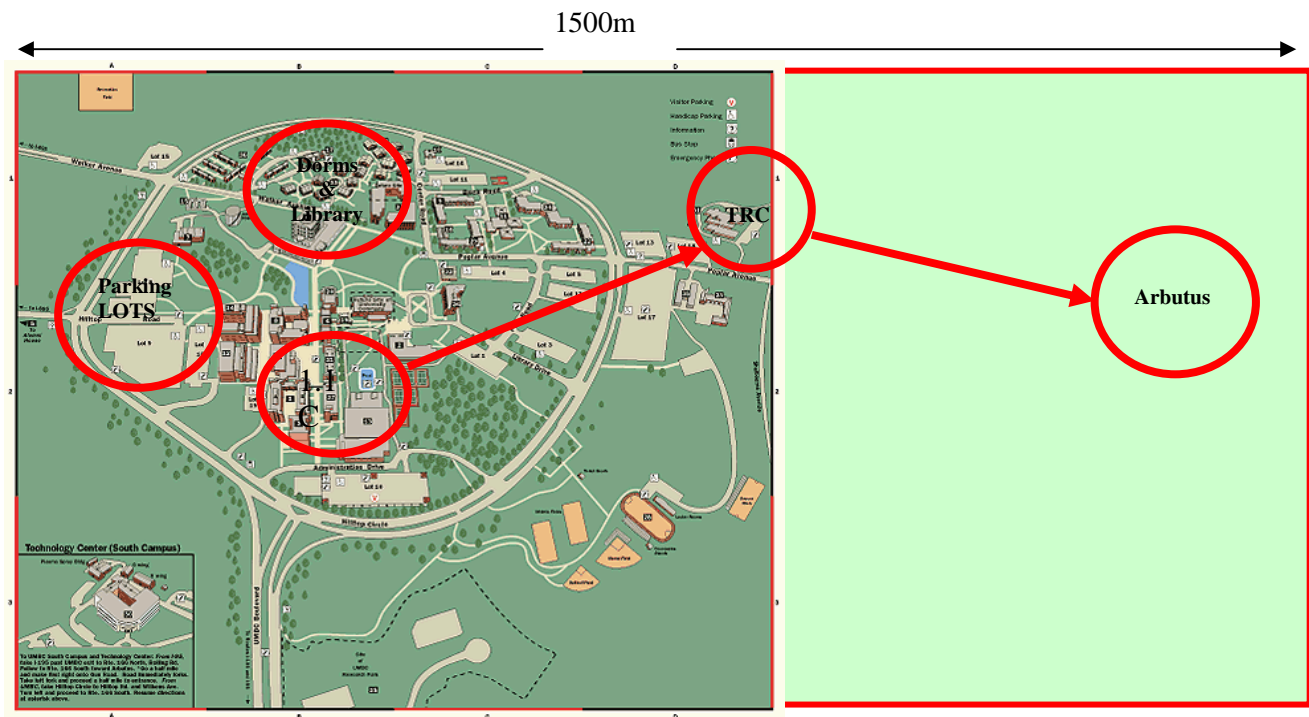


Fig 12: The mobile node movement scenario in UMBC Campus. Network is partitioned if there is no meeting in the TRC (Technology Research Center) building.

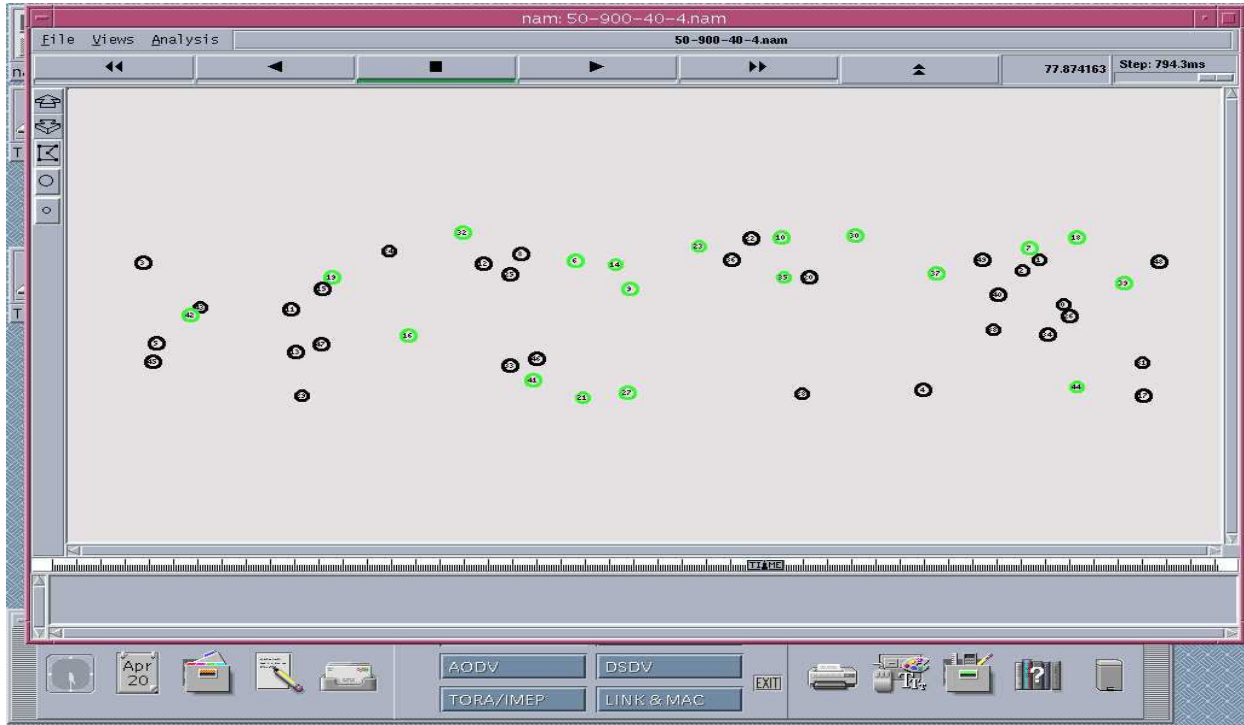


Figure 13 : Snapshot of a typical movement scenario observed in random waypoint model with 50 nodes in ns simulator.

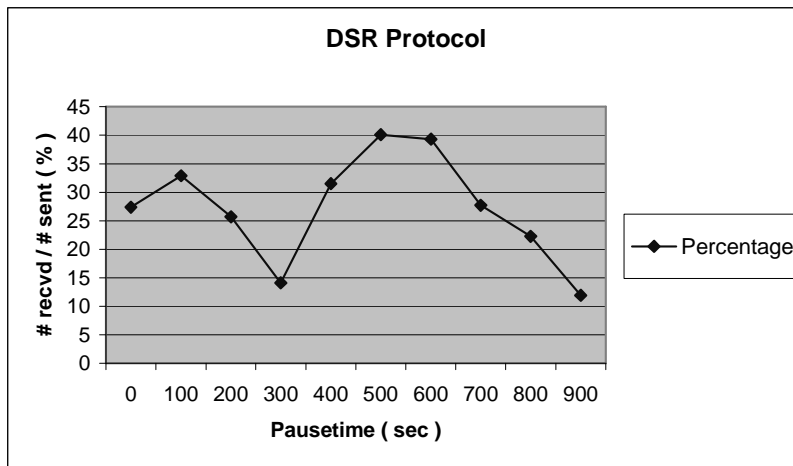


Figure 14 : The effect of movement patterns and partitions on throughput in DSR.

Fig. 15 shows one small changes we have made to the scenario generator in ns simulator to obtain the UMBC Model. Instead of using the totally random destination selector in the original code shown in 15a, we select to move the nodes between UMBC buildings as shown in 15b.

```

void Node::RandomDestination()
{
    destination.X = uniform() * MAXX;
    destination.Y = uniform() * MAXY;
    destination.Z = 0.0;
    assert(destination != position);
}

```

(a)

```

#define PARK      0
#define ECS       1
#define UC        2
#define LIB       3
#define ERICKSON  4
#define TRC       5
#define ARBUTUS   6

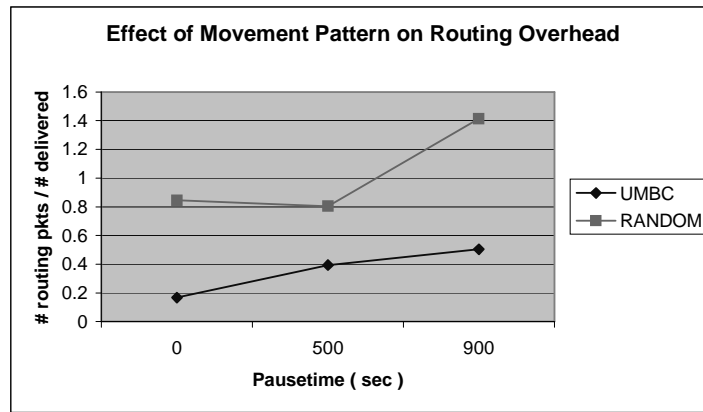
void Node::SelectUMBCBuilding()
{
    ...
    choice = random() % 7;
    switch( choice ){
        case PARK:
            destination.X = 100;
            destination.Y = 300;
            break;
        case ECS:
            destination.X = 200;
            destination.Y = 250;
            break;
        case UC:
            destination.X = 300;
            destination.Y = 200;
            break;
        case LIB:
            destination.X = 300;
            destination.Y = 450;
            break;
        ...
        default:
            printf("UNKNOWN LOCATION");
            break;
    }
}

```

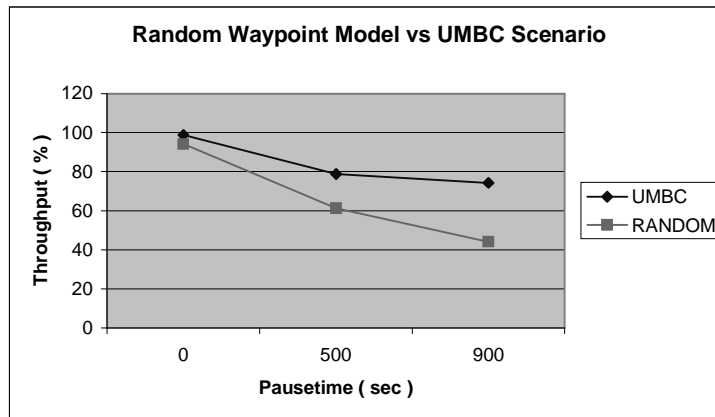
(b)

Figure 15: Modifications to ns scenario generator to support UMBC movement scenario.

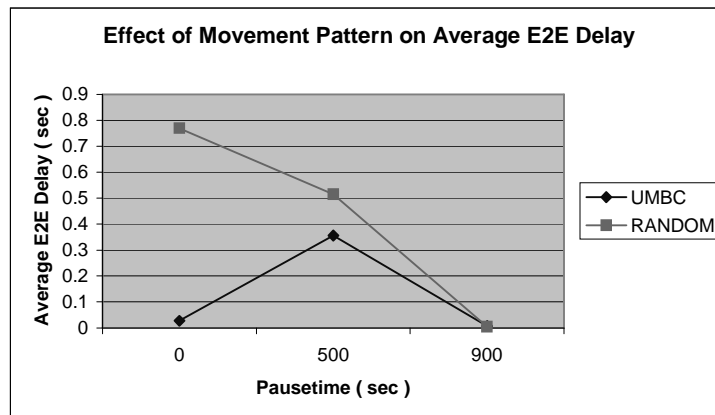
Fig. 16a shows that in UMBC scenario most of the nodes were packed up together around the UC (University Center) as deduced from the higher throughput and not as many partitions as in the random waypoint model were observed in UMBC scenario. So, in this case results give the credit to random waypoint model by proving that this model could also include different kinds of topological cases and it is neither a smooth nor an unrealistic assumption as we have argued before in this section. Fig 16b shows that UMBC model required less routing than a totally random movement because of its more deterministic nature. Fig 16c is just an extension to these comments with average end-to-end delay being less in UMBC model than in random model. UMBC movements were more predictable and tractable by the ad-hoc protocols (specifically DSR in this case) compared the random movement model.



(a)



(b)



(c)

Figure 16 : Comparison of Random Waypoint Model with UMBC scenario.

7 Conclusions

We compared four routing protocols DSR, AODV, DSDV and CEDAR based on their throughput, routing overhead and average end-to-end data packet delivery delays under a wide range of simulated network conditions. We have changed the pausetimes, speeds, bit-rates, traffic generators, area sizes, movement scenarios and many other parameters details of which can be found in Section V.

We have most generally found that in highly dynamic ad-hoc environments, knowing as much as possible about what is happening around, listening to other nodes, caching as much information as possible and thus keeping alternatives whether they are most up to date or not is a plus. This increases the responsiveness of

the routing protocol, which is very much valued in real-time environments. We have observed that being a very good composer of all the responsiveness ideas, DSR (Dynamic Source Routing) protocol was quite successful under various conditions. Other protocols were usually trading routing overhead or average delay with throughput. DSDV did not increase the number of routing packets and this caused degradation in throughput. AODV managed throughputs as high as DSR, but with higher routing overhead and delay.

Some of the other experiences and proposed future work may be found in Appendix A.

Acknowledgements

We would like to thank our advisor Dr. Anupam Joshi for initiating this effort and contributing with his valuable comments. We also credit TIMELY group in UIUC for providing us with their CEDAR implementation.

REFERENCES

- [1] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A Media Access Protocol for Wireless LANs." ACM Sigcomm '94, London, UK. September 1994.
- [2] D. Johnson and D. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks" chapter5, pp.153-179, Mobile Computing. T. Imelinski and H. Korth, eds. Kluwer academic Publishers.
- [3] J. Broch, D. Johnson and D. Maltz, "The Dynamic Source Routing Protocol for mobile ad hoc networks", <http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-03.txt>, October 1999. IETF Internet Draft.
- [4] C. Perkins, E. Royer, S. Das. "Ad Hoc On-Demand Distance Vector (AODV) Routing", <http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-05.txt>, March 2000, IETF Internet Draft.
- [5] V. Park, S. Corson, "Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification", <http://www.ietf.org/internet-drafts/draft-ietf-manet-tora-spec-02.txt>, October 1999, IETF Internet Draft,
- [6] S. Lee, W. Su, M. Gerla, "On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks", <http://www.ietf.org/internet-drafts/draft-ietf-manet-odmrp-02.txt>, January 2000, IETF Internet Draft.
- [7] P. Sinha, R. Sivakumar, V. Bhargavan, "CEDAR: A Core Extraction Distributed Ad-Hoc Routing Algorithm".
- [8] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98) , pages 85-97, October 1998
- [9] S. Das, C. Perkins and E. Royer, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks"
- [10] D. Maltz, J. Broch, J. Jetcheva and D. Johnson. "The Effects of On-Demand Behaviour in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks", IEEE Journal on Selected Areas in Communication, 1999.
- [11] S. Corson and J. Macker, " Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", RFC 2501 MANET. January 1999

[12] K. Fall and K. Varadhan, "ns notes and Documentation", 1999,
<http://www-mash.cs.berkeley.edu/ns/>

[13] CMU Monarch Project, "The CMU Monarch Project's Wireless and Mobility Extensions to ns",
August 1999, <http://www.monarch.cs.cmu.edu>

Appendix A. Experiences with the NS Simulator

In this section we would like to note a few things about our experiences with ns network simulator with mobility extensions by Monarch group in CMU as a future reference to the interested :

- Relatively easy to download, install and start running initial tests when compared with other network simulators. The initial cost of learning is has been decreased with the availability of detailed documentation, handy tips and ready to run examples.
- Simulator has a wide range of library support in networking and is expanding with everyday additions.
- Most wireless simulations will have huge processing power (CPU), memory and disk space requirements. Obtaining one of the points in the above graphs, for example: a 50 node ad-hoc routing simulation with 40 connections, may take around half an hour on a SUN Ultra 5 workstation with 128Mbytes of memory and dump 5-10MBytes trace files. This fact has also been mentioned in David Johnson's paper.[2]
- Excel is insufficient for parsing the trace files, since rows are limited to 65536 rows. A typical trace file contains 100,000 lines of dumped information. Scripting languages like Perl were found to be much more practical parsing the dumped trace files for multiple performance metrics.
- CEDAR code we obtained from TIMELY group in UIUC unfortunately did not compile due to architectural dependencies. We set adaptation of this code into our simulator as future work.
- We detected that AODV protocol may go into infinite loops with high number of mobile nodes and high connectivity values.