AN ADAPTIVE DISTANCE VECTOR ROUTING ALGORITHM

FOR MOBILE, AD HOC NETWORKS

APPROVED BY SUPERVISING COMMITTEE:

Dr. Rajendra V. Boppana, Chair

Dr. Robert E. Hiromoto

Dr. Richard F. Sincovec

Accepted:

Dean of Graduate Studies

AN ADAPTIVE DISTANCE VECTOR ROUTING ALGORITHM FOR MOBILE, AD HOC NETWORKS

by

SATYADEVA PRASAD KONDURU, B.Tech.

THESIS
Presented to the Graduate Faculty of
The University of Texas at San Antonio
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences and Engineering
Division of Computer Science
August 2000

Acknowledgments

My thanks go to my adviser, Professor Rajendra V. Boppana, for his guidance throughout this thesis work. His insights and the discussions we had have been very valuable for this thesis and for myself. I am also grateful to other faculty members and colleagues for their suggestions.

This thesis work has been partially supported by DOD/AFOSR grant F49620-96-1-0472.

August 2000

AN ADAPTIVE DISTANCE VECTOR ROUTING ALGORITHM FOR MOBILE, AD HOC NETWORKS

Satyadeva Prasad Konduru, B.Tech.
The University of Texas at San Antonio, 2000

Supervising Professor: Dr. Rajendra V. Boppana

In this thesis, we investigate routing algorithms that combine appropriate proactive and on-demand routing techniques for mobile, ad hoc networks (MANETs). In particular, we present an Adaptive Distance Vector (ADV) routing algorithm, which is a proactive technique, that actually outperforms two extensively studied on-demand routing algorithms Ad hoc On-demand Distance Vector (AODV) and Dynamic Source Routing (DSR), especially in high mobility situations. ADV exhibits some on-demand characteristics by maintaining routes only to the destinations that are part of active connections and adaptively changing the frequency of the routing updates by the use of load- and mobility-based criteria.

We have compared ADV with AODV and DSR, and analyzed the performance under various network conditions using the widely used packet-level simulator *ns-2* with wireless extensions. Our simulations indicate that ADV outperforms AODV and DSR especially in high mobility cases by giving significantly higher (50% or more) peak throughputs and lower packet delays. Furthermore, ADV uses fewer routing and control overhead packets than that of AODV and DSR especially at moderate to high loads. But ADV still needs some fine tuning in order to perform better than AODV and DSR at low speeds and infrequent node movements (different pause times) in the network. Also, the load-based adaptive criteria need to be further improved to handle very low data rates. But, when considered in totality, ADV has all the characteristics to be an effective routing protocol for the ad hoc networks.

We have also enhanced AODV, an on-demand algorithm, with a proactive technique aimed at reducing packet latencies in high speed networks. The technique involves broadcasting receiver-initiated "beacon" packets periodically, which would help maintain fresh routes to the corresponding receivers at

all other nodes in the network. AODV with this proactive technique reduces packet latencies by a factor of 2 to 3 at low to medium packet rates in high speed networks. The routing packets transmitted are also reduced as the beacons obviate the need for some route discovery processes. The throughput obtained is about the same as that in original AODV.

Our results indicate the benefits of combining both proactive and on-demand routing techniques in designing suitable routing protocols for MANETs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A mobile, ad hoc network (MANET) is an autonomous system of mobile hosts connected by wireless links. With recent performance advancements in computer and wireless communications technologies, advanced mobile wireless computing is expected to see increasingly widespread use and application, much of which will involve the use of the Internet Protocol (IP) suite. So the area of networking support for MANETs, aimed at incorporating robust and efficient routing functionality into the mobile nodes, is getting much attention.

An added advantage/disadvantage for a mobile, ad hoc network is that it can be formed without the aid of any centralized administration or standard support services. The advantage is that the mobile nodes can dynamically form a network of their own "on the fly" without any need for wired infrastructure. The possible situations can arise in emergency disaster relief operations or soldiers relaying information for situational awareness on the battlefield. The disadvantage is that, two hosts that may wish to exchange data might not be able to communicate directly due to limited propagation range of each mobile hosts' wireless transmissions. It may be necessary for one mobile host to enlist the aid of others in forwarding a packet to its destination. Hence efficient routing algorithms to create, maintain and repair routing paths are necessary in such environments.

Conventional routing protocols developed for traditional wired LANs/WANs may be used for routing in ad hoc networks, treating each mobile host as a router. Such algorithms broadly come under the category of *proactive* algorithms since routing information is disseminated among all the nodes in the

network through out the network operating time irrespective of the need for any such route. Some of the latest approaches, called *on-demand* algorithms, discover and maintain routes only on a need basis. Many performance comparisons done till now have shown that on-demand algorithms perform better than proactive algorithms. So on-demand algorithms have been claimed as better suited for mobile and ad hoc environments. But we find that a combination of proactive and on-demand techniques perform better than either approach alone. This thesis work takes a second look at the proactive algorithms and aims at bringing attention to the conventional routing techniques and their benefits.

1.1 Routing Issues

This section focuses on special challenges to MANET routing protocols, desirable properties of such routing protocols and the applicability of conventional routing techniques in MANET environments.

1.1.1 Special challenges to MANET routing protocols

The routing algorithms for MANETs must cope with some challenges that are specific to the mobile network environment. Some of them are listed below:

Lack of default router The routing algorithm should be simple and efficient in its resource usage so that even inexpensive and low-powered mobile nodes can run it.

Frequent changes in network topology due to both node mobility and temporary losses of wireless channels The routing algorithm must be highly adaptive to mobility and topology changes so that it provides predictable performance under transient and steady-state conditions.

Low bandwidth channels The routing algorithm should minimize the number of hops taken by packets, and minimize overhead in collecting and disseminating routing information with other nodes.

Unreliable broadcasts Since a broadcast is the primary mechanism by which route discovery and propagation of routing information is done, the routing algorithm must be robust and should not display erratic behavior if packets carrying routing information are lost.

1.1.2 Desirable properties of routing protocols

This section gives some desirable properties that any MANET routing protocols should possess in order to incorporate robust and effective routing functionality into the network.

Distributed operation The protocol should of course be distributed. It should not be dependent on a centralized controlling node. This is the case even for stationary networks. But the difference is that the nodes in an ad hoc network can enter/leave the network very easily and because of mobility the network can be partitioned.

Loop free To improve the overall performance, we want the routing protocol to guarantee that the routes supplied are loop-free. Even if loops are formed they should be temporary in nature and should be eliminated over a short period of time. This avoids any waste of bandwidth or CPU consumption.

Unidirectional link support The radio environment can cause the formation of unidirectional links. Utilization of these links in addition to bidirectional links improves the routing protocol performance.

Security The radio environment is especially vulnerable to impersonation attacks. So as to ensure the wanted behavior from the routing protocol, we need some sort of preventive security measures. Authentication and encryption is probably the way to go and the problem here lies within distributing keys among the nodes in the ad hoc network.

Power conservation The nodes in an ad hoc network can be laptops and thin clients, such as PDAs that are very limited in battery power and therefore uses some sort of stand-by mode to save power. It is therefore important that the routing protocol has support for these sleep-modes.

Multiple routes To reduce the number of reactions to topological changes and congestion, multiple routes can be used. If one route has become invalid, it is possible that another stored route could still be valid and thus saving the routing protocol from initiating another route discovery procedure.

Quality of service support Some sort of Quality of Service support is probably necessary to incorporate into the routing protocol. To support real-time traffic the routing algorithm should facilitate Quality of Service implementation.

None of the already proposed protocols for MANET have all the desirable properties, but it is necessary to remember that the protocols are still under development and are probably extended with more functionality.

1.1.3 Current routing solutions

There are two types of routing protocols suggested for MANETs - proactive and on-demand. This section describes only the underlying algorithms on which either types of protocols are based. The routing protocols, specifically designed for MANETs are described in detail in Chapter 2.

Proactive routing algorithms

A natural method for trying to provide routing in an ad hoc network is to simply treat each mobile host as a router and to run a conventional routing protocol among them based on either *distance vector* or *link-state*. These are so called proactive routing algorithms.

Distance Vector In *distance vector* routing, every router in the network maintains a routing table in which all of the possible destinations within the network and the number of hops to each destination are maintained. Each router periodically broadcasts this information to each of its neighbor routers, and uses the values received for each destination from its neighbors to update the values for its own table. By comparing the distances received for each destination from each of its neighbors, a router can determine which of its neighbors is the correct “next hop” on the shortest path toward each destination and thereby forwards to it any data packet meant for that destination.

It is well known that distance vector can cause the formation of both short-lived and long-lived loops [13]. The primary cause for this is that the nodes choose their next-hops in a completely distributed manner based on information that can be stale. By transmitting routing table updates more frequently such as when any information in the table changes, the algorithm converges more quickly to the correct path (for example, when a link comes up or down), but the overhead in CPU time and network bandwidth

for transmitting routing updates increases. One distance vector routing protocol that has already been proposed for MANETs is the Destination Sequenced Distance Vector (DSDV) protocol [27].

Link-State In *link-state* routing, each node maintains a complete picture of the topology of the entire network. Each router monitors the cost of the link to each of its neighbor routers, and periodically broadcasts an update of this information to all other routers in the network. Given this information of the cost of each link in the network, each router computes the shortest path to each possible destination. When presented a packet for forwarding to some destination, each router forwards the packet to the next hop router based on its current best path to that destination.

Link-state routing protocols converge much more quickly as conditions in the network change, but generally require more CPU time (to compute the complete shortest path to each possible destination) and more network bandwidth (to broadcast the routing update from each router to all other routers in the entire network) than distance vector algorithms. One link-state routing protocol that has already been proposed for MANETs is the Optimized Link State Routing (OLSR) protocol [21].

On-demand routing algorithms

The second type of routing algorithms, called the on-demand routing algorithms, learn and maintain routing paths only when there are packets to be transmitted. There are two such protocols that are extensively studied in the literature [1, 7, 14]. One is based on the source routing concept and the other is again based on the distance vector approach.

Source Routing *Source routing* means that each packet must carry the complete path that the packet should take through the network. The routing decision is therefore made at the source. The advantage with this approach is that it is very easy to avoid routing loops. Also the intermediate nodes need not keep route information because the path is explicitly specified in the data packet. The disadvantage is that each packet requires a slight overhead. Dynamic Source Routing (DSR) [8] is one such protocol designed for MANETs.

Distance Vector The Ad hoc On-demand Distance Vector (AODV) [9] protocol is an on-demand protocol that is based on the distance vector routing technique. But the difference is that, unlike the proactive distance vector algorithms, AODV requests for a route only when needed and does not maintain routes to destinations that are not actively used in communications. As long as the sources of the communication connections have valid routes to the corresponding destinations, AODV does not play any role.

1.2 Contributions of this thesis

It has been claimed until now that on-demand algorithms, like Ad hoc On-demand Distance Vector (AODV) and Dynamic Source Routing (DSR), are superior to proactive algorithms, like Destination Sequenced Distance Vector (DSDV), that are based on traditional routing algorithms [1, 11, 14]. But proactive techniques do improve performance when combined with some on-demand characteristics. This thesis report presents a version of distance vector routing that is adaptive to network load and mobility. The adaptive criteria show some on-demand characteristics by varying the frequency and the size of the routing updates according to the network conditions. We have done extensive simulations to compare the Adaptive Distance Vector (ADV) protocol with two on-demand protocols AODV and DSR and found that ADV outperforms the on-demand protocols especially when nodes move at high speeds. The peak throughput achieved is 50% more than the on-demand protocols. Also ADV offers lower packet latencies and transmits fewer routing overhead packets. Thus ADV can be an effective alternative to the on-demand protocols.

We also incorporate a proactive routing technique in AODV which involves broadcasting receiver-initiated beacons periodically, thereby refreshing the routes to the corresponding receivers at all the nodes in the network. In highly mobile networks, this proactive beacon technique not only reduces packet latencies by a factor of 2-3 but also reduces the routing packets transmitted, while maintaining the throughputs. However it is not effective in low speed networks and incurs extra routing overhead.

The main contributions of this thesis are i) designing a proactive routing algorithm with some on-demand characteristics to make it responsive to the network load and mobility conditions, and ii) enhancing an on-demand algorithm with a proactive technique,to reduce latencies in high mobility situations. In both cases, we demonstrate that a mix of proactive and on-demand techniques improve performance of routing algorithms.

1.3 Organization of the report

The rest of the thesis report has been organized as follows: Chapter 2 gives the background on the work that has already been done on mobile ad hoc networks. It describes the routing protocols that have already been proposed, the underlying media access layer (MAC) and the simulation environment. Chapter 3 describes in detail the new Adaptive Distance Vector routing protocol that we have developed and discusses the related design and implementation issues. It also describes a small enhancement done on AODV with a view to reduce the average packet latency. Chapter 4 gives the set of experiments and performance comparisons that had been done on various routing protocols and their enhancements. Chapter 5 summarizes the work done.

Chapter 2

Related Background Work

In this chapter, we first give an overview on the distance vector routing technique and its derivative that have already been proposed for interconnection networks. Next we describe an ad hoc routing protocol that is based on the distance vector routing. Later we describe two of the well known on-demand ad hoc routing protocols. The last section gives a brief overview of the Network Simulator (ns-2) [4] that is used to compare the performances of various routing protocols. To simulate the mobile wireless radio environment we have used a mobility extension to ns-2 that is developed by the CMU Monarch project at Carnegie Mellon University.

2.1 Distance Vector Routing

The Routing Information Protocol (RIP) [20] used in interconnection networks is a distance vector (DV) routing algorithm. In distance vector routing, each router maintains a routing table giving the distance from itself to all possible destinations. Each routing table entry consists of destination IP address, the distance to that destination (metric) and next hop neighbor which forwards any data packet to that destination. These entries are also given in the Table 2.1 below. Initially the routing table entries corresponding to neighbor nodes have a distance of 1 and all others an invalid distance metric of infinity. Each router periodically broadcasts this table information to each of its neighbor routers, and uses the values received for each destination from its neighbors to compute updated values for its own table. By comparing the distances received for each destination from each of its neighbors, a router can determine

which of its neighbors is the correct “next hop” on the shortest path toward each destination. When presented a packet for forwarding to some destination, each router simply forwards the packet to the correct next hop router.

Routing table entry	Function
Destination	IP address of the destination node
Hop	Next node in the path to destination
Metric	Number of hops to reach the destination

Table 2.1: Fields in a routing table entry at a node in Distance Vector routing.

This is the classical Distributed Bellman-Ford (DBF) algorithm [22]. It is well known that DV can have both short-lived and long-lived routing loops. The primary cause for formation of routing loops is that nodes choose their next-hops in a completely distributed fashion based on information which can possibly be stale and, therefore, incorrect. A simple example that illustrates a routing loop involves two nodes, *a* and *b*. The scenario represents something like this: *a* thinks that it has a route to a destination, say *x*, through *b* and *b* thinks it has a route to the destination *x* through *a*. When transmitting a packet to the destination *x*, *a* sends it to *b* to be forwarded. But *b* again sends it back to *a* thinking that *a* is the forwarding node. This process goes on back and forth until the loop is broken or the maximum hop count of the data packet is reached. Routing loops can be eliminated by forcing all nodes in the network to participate in some form of internodal coordination mechanism which is effective when topological changes are rare.

RIP handles routing loops by using split horizon with poisoned reverse technique and transmitting triggered updates. Split horizon is a scheme for avoiding problems caused by including routes in updates sent to the routers from which they were learned. The simple split horizon scheme omits routes learned from one neighbor in updates sent to that neighbor. Split horizon with poisoned reverse includes such routes, but set set their metrics to infinity. However this prevents any routing loops that involve only two routers. It is still possible to end up with patterns in which three routers are engaged in mutual deception. For example, A may believe it has route through B, B through C, and C through A. Triggered updates are an attempt to break such loops. Whenever a router changes the metric for a route, it is required to send

update messages almost immediately, even if it is not yet time for one of the regular update message. Unfortunately this method fails to remove the counting-to-infinity problem completely. It is possible that after a triggered update has gone through a node, it might receive a normal update from one of the nodes that hasn't yet gotten the word about the faulty route. This could reestablish an orphaned remnant of the faulty route. Of course, this is very unlikely if triggered updates happen quickly enough. Though RIP is extensively used in small intranets, its usefulness within an ad hoc environment is limited since it is not designed to handle rapid topological changes.

2.2 A Distance Vector Algorithm for MANETs

This section describes an ad hoc routing protocol, called Destination Sequenced Distance Vector (DSDV) [27], that is based on the distance vector algorithm. This protocol has been designed based on the simple routing methodology of RIP, and avoids the looping problem.

Avoiding Loops DSDV [27] solves the looping problem by attaching sequence numbers to routing entries. A node increments its current sequence number and includes it only in its periodic updates. Any node that invalidates its entry to a destination because of loss of next hop node, will increment the sequence number and uses the new sequence number in its next advertisement of this route (in full updates only). A node invalidates or modifies its routing entry if a neighbor broadcasts a routing entry to the same destination with a higher sequence number. An invalidated entry can only become valid by the routing information propagated by the destination node with a higher sequence number.

In DSDV, as in other distance vector algorithms, each node maintains a table of routing entries, with each entry indicating destination node number, number of hops required to reach the destination, and the next node in the path to the destination. The routing table entries in all the nodes for a given destination collectively specify a *virtual destination-based tree* to send packets to that destination. A simplistic view of DSDV is that it maintains one such destination tree for each node in a distributed manner. Initially the routing tables do not contain valid (finite distance) entries to all nodes in the network. At

periodic intervals, nodes broadcast their routing tables to their neighbors. A node incorporates any routing table information received from its neighbor into its routing table by comparing the entries and picking the routes with better hop counts. These periodic updates to routing entries are done for all paths, even for those that may not be in use. If three periodic updates from a node are missed then the link to that node is declared broken. Alternatively, MAC link layer can report on lost neighbors when a presumed neighbor does not respond to a request-to-send (RTS) packet (RTS packet is used in reserving the wireless medium for the transmission of any unicast packet, explained in detail in Section 2.5). When a node detects the loss of a neighbor, it invalidates (by setting the hop metric to infinity) all of its routing entries that have the lost neighbor as the next node and broadcasts the changed entries to its neighbors. These triggered updates to routing tables may result in invalidation of routing entries in its neighbors and additional triggered updates by them.

DSDV was shown to have very high routing overhead compared to other on-demand routing protocols [1]. Though the number of routing packets transmitted per second are less in DSDV, the large number of routing entries in the update packets account for the large overhead. Most of these entries can be accounted for by the route improvements (like better sequence numbers) propagated to maintain fresh routes to all the nodes in the network. Furthermore, triggered updates are likely to invalidate too many routing entries needlessly.

To see this, consider the virtual destination-tree for some node, say x. When a node, say y, loses its path to x, it invalidates its routing entry for x by incrementing the sequence number associated with the route and advertises it to its neighbors. Neighbors of y in turn invalidate their entries to x without regard to whether they need to use y to reach x or not and propagate the invalid route to their neighbors. This continues until node x responds with a higher sequence number. Such route invalidations can destroy the part of the destination tree that is unaffected by y's loss of path to x, and cause needless triggered updates.

Fix for propagation of needless triggered updates. It is easy to contain the impact of invalid route propagation by specifying that a node may invalidate its routing entry based on a neighbor's update only if the neighbor's entry has a higher sequence number and the neighbor is currently the specified next hop. With this fix, the invalidations are propagated only to the nodes in the subtree of the virtual destination-based tree, that are affected by the link failure. However, a routing entry may be modified based on neighbor's entry if the neighbor advertises a valid route with a higher sequence number or better metric with the same sequence number as in DSDV.

2.3 On-demand Routing Algorithms for MANETs

In this section we present in detail two on-demand routing algorithms that are widely studied [1, 7, 14]; namely, Ad hoc On-demand Distance Vector Protocol (AODV) and Dynamic Source Routing Protocol (DSR). In the last subsection we briefly outline various other algorithms that are proposed for MANETs. Most of these protocols are hybrid proactive/on-demand type protocols.

2.3.1 Ad hoc On-demand Distance Vector - AODV

The Ad hoc On-demand Distance Vector (AODV) [9] routing protocol enables multi-hop routing between participating mobile nodes wishing to establish and maintain an ad hoc network. AODV is based upon the distance vector algorithm. The difference is that AODV is reactive, as opposed to proactive protocols like DSDV, i.e. AODV requests a route only when needed and does not require nodes to maintain routes to destinations that are not actively used in communications. As long as the the sources have valid routes to their corresponding destinations, AODV does not play any role.

Route discovery A node broadcasts a RREQ when it determines that it needs a route to a destination and does not have one available. This can happen if the destination is previously unknown to the node, or if a previously valid route to the destination expires. To prevent unnecessary broadcasts of RREQs, the source node uses an expanding ring search technique as an optimization. In an expanding ring search,

the a source node initially uses a time-to-live (TTL) = TTL_START in the RREQ packet IP header and sets a timeout for receiving a reply (RREP). Upon timeout the source retransmits with TTL incremented by TTL_INCREMENT. This continues until TTL reaches TTL_THRESHOLD beyond which a TTL = NET_DIAMETER is used for each rebroadcast. Nodes receiving RREQs set up reverse paths to sources of RREQs in their routing tables, and either reply to the RREQ if they already have an entry for the destination in question or forward the RREQ. The route is made available by unicasting a route reply (RREP) back to the source. The source receives at least one RREP (from the destination in the worst case) for its RREQ.

Route maintenance Every routing table entry maintains a route expiry time which indicates the time until which the route is valid. Each time that route is used to forward a data packet, its expiry time is updated to be the current time plus ACTIVE_ROUTE_TIMEOUT. An existing route entry may be invalidated if it is unused within such specified time interval or the next hop node is no longer a viable node to reach the destination. In case a routing entry is invalidated, the invalidation is propagated to neighbors that have used this node as their next hop. AODV requires the nodes to exchange hello messages periodically with their neighbors or receive feedback from the link layer when a loss of a neighbor is detected. When a node detects that a route to a neighbor is no longer valid, it will remove the routing entry and send a link failure message. The latter is a triggered route reply message to the neighbors that are actively using the route, and informs them that this route no longer is valid. For this purpose AODV uses an active neighbor list to keep track of the neighbors that are using a particular route. The nodes that receive this message will repeat this procedure. The message will eventually be received by the affected nodes that can either choose to stop sending data or request a new route by sending out a new RREQ.

2.3.2 Dynamic Source Routing - DSR

A routing entry in DSR contains all the intermediate nodes to be visited by a packet rather than just the next hop information maintained in DSDV and AODV. A source puts the entire routing path in the data packet, and the packet is sent through the intermediate nodes specified in the path (similar to the IP strict source routing option [16]). If the source does not have a routing path to the destination then it performs a route discovery by flooding the network with a route request (RREQ) packet. The RREQs record route information as they visit intermediate nodes on the way to the destination. Any node that has a path to the destination in question can reply to the RREQ packet by sending a route reply (RREP) packet. The reply is sent using the route recorded in the RREQ packet. A node that receives a RREQ can use the path recorded to improve its path to the source. To reduce the cost of route discovery, each node maintains a cache of source routes it has learned or overheard, that it aggressively uses to limit the frequency and propagation of RREQs. When an intermediate nodes discovers that a source route is broken, the source node is notified with a route error (RERR) packet. The source node can then attempt to use any other route to the destination that already is in its cache or can invoke route discovery again to find a new route.

DSR specific optimizations To limit the need for route discovery, DSR also allows nodes to operate their network interfaces in promiscuous mode and snoop all (including data) packets sent by their neighbors. Since complete paths are indicated in data packets, snooping can be very helpful in keeping the paths in the route cache fresh. So DSR attempts to exploit the shared channels better than AODV or DSDV. To further reduce the cost of route discovery, the RREQs are initially broadcasted to neighbors only (1-hop limit), and then to the entire network if no reply is received. Another optimization feasible with DSR is the gratuitous route replies; when a node overhears a packet containing its address in the unused portion of the path in the packet header, it sends the shorter path information to the source of the packet. Also, an intermediate node may replace a packet's current path specification with an alternate path if it is unable to send the packet to the original next hop node.

DSR specification and implementation [2] contains the above mentioned optimizations. Of these, the route replacement and gratuitous route replies are specific to DSR. Another optimization specific to DSR is snooping and learning better paths. This requires a mobile node to continually process all packets being transmitted within its listening area. This may not be a feasible optimization for low-powered devices.

2.4 Other algorithms for MANETs

There are many other routing algorithms proposed for MANETs. Some of them are described below.

Optimized Link State Routing Protocol - OLSR

The optimized link state routing protocol (OLSR) [21] is a proactive routing protocol. The protocol inherits the stability of a link state algorithm and has the advantage of having the routes immediately available when needed due to its proactive nature. To reduce the size of the control packets, it propagates only a subset of links with its neighbors who are its multipoint relay selectors. Also it minimizes flooding of its control traffic by using only the selected nodes, called multipoint relays, to diffuse its messages. The nodes in a multipoint relay set of a node are arbitrarily selected from among its one hop neighbors in such a manner that the set covers (in terms of radio range) all the nodes that are two hops away.

Apart from its normal periodic control messages, the protocol does not generate extra control traffic in response to link failures and additions. Thus it is suitable for networks with a high rate of topological changes.

Temporally Ordered Routing Algorithm - TORA

The temporally ordered routing algorithm (TORA) [18] is designed to minimize reaction to topological changes and hence control messages are typically localized to a very small set of nodes. It guarantees that all routes are loop-free (temporary loops may form) and typically provides multiple routes to destinations to cope up with changes in network topologies. Route optimality is considered of secondary

importance, and longer routes are often used to avoid the overhead of discovering newer routes. TORA is layered on top of IMEP, the Internet MANET Encapsulation Protocol, that is required to provide reliable, in-order delivery of all routing control messages from a node to each of its neighbors, link status sensing etc. However, this algorithm is sensitive to routing packet losses and has been shown to perform poorly compared to AODV and DSR in previous studies [1, 10].

Zone Routing Protocol - ZRP

The zone routing protocol (ZRP) [23] is a hybrid on-demand/proactive routing protocol. ZRP views a MANET as a collection of zones and uses a proactive protocol within a zone. Each zone has a zone leader which also runs a reactive protocol for maintaining routing tables and information across the zones. Routes can be found very fast within the routing zone, while routes outside the zone can be found by efficiently querying selected nodes in the network.

A routing zone is defined as a set of nodes, within a specific minimum distance in number of hops from the node in question. The operation of ZRP can be adjusted to any network operational conditions by changing the routing zone diameter. However this is not done dynamically, but instead set up before the network is operational. The performance of this protocol may depend quite a lot on this decision.

Core-Extraction Distributed Ad hoc Routing - CEDAR

An opposite approach is taken by the CEDAR algorithm [25] where a collection of minimum dominating set of nodes form a *core* and a proactive algorithm is run among the nodes in the core, and a reactive algorithm to reach all other nodes. The other key components of CEDAR involve propagating the bandwidth availability information of stable links in the core subgraph and the Quality of Service (QoS) route computation algorithm. (R. Sivakumar et. al. [25] considers only bandwidth as the QoS parameter.)

Robustness, rather than optimality, is the primary concern of CEDAR. Like all on-demand algorithms, route computation is only performed when a route is requested. The QoS algorithm is robust

and uses only local state for route computation at each node. Hence CEDAR does not require high maintenance overhead even for highly dynamic networks.

Cluster Based Routing Protocol - CBRP

Cluster Based Routing Protocol (CBRP) [24] divides the nodes into a number of overlapping or disjoint clusters in a distributed manner (differs from ZRP where the zones are all overlapping). Inter-cluster routes are discovered dynamically using the cluster membership information kept at each cluster head.

The clustering approach is very good when dealing with large ad hoc networks. This solution is more scalable than the other protocols because it uses a clustering approach that limits the number of messages to be sent. The size of each cluster is an important parameter to how well the protocol behaves.

Associativity Based Routing - ABR

Associativity-based routing (ABR) protocol [26] is a source-initiated on-demand protocol in which a route is selected based on nodes having associativity states that imply periods of spatial, temporal, connection and signal stability. The selected routes are likely to be long-lived and hence require infrequent restarts, resulting in higher achievable throughput.

Route selection is based primarily on the aggregated degree of associativity of nodes along the path. Hence, although the resulting path does not necessarily result in the smallest possible number of hops, the path tends to be longer-lived than other routes. ABR, however, relies on the fact that each node is beaconing periodically. The beaconing interval must be short enough so as to accurately reflect the spatial, temporal and connectivity state of the mobile hosts. This beaconing requirement may result in additional power consumption.

2.5 MAC 802.11

An implementation of the IEEE 802.11 Media Access Control (MAC) [6] protocol is used as an underlying media access protocol. The MAC layer handles collision detection, fragmentation and acknowl-

edgements. This protocol may also be used to detect transmission errors. 802.11 is a CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) protocol. It avoids collisions by checking the channel before using it. If the channel is free, it can start sending. If not it must wait a random amount of time before checking again. For each entry an exponential backoff algorithm will be used. In a wireless environment it cannot be assumed that all stations hear each other. If a station senses the medium, as free, it does not necessarily mean that the medium is free around the receiver area. This problem is known as the *hidden terminal problem*. To overcome these problems the Collision Avoidance mechanism together with a positive acknowledgement scheme is used. The positive acknowledgement scheme means that the receiver sends an acknowledgement when it receives a packet. The sender will try to transmit this packet until it receives the acknowledgement or the number of transmits exceeds the maximum number of transmits.

The transmission of each unicast packet is preceded by a Request-to-Send/Clear-to-Send (RTS/CTS) exchange that reserves the wireless channel for transmission of a data packet. Each correctly received unicast packet is followed by an Acknowledgement (ACK) to the sender, that retransmits the packet a limited number of time until this ACK is received. Broadcast packets are sent only when carrier sense indicates that the medium is clear but they are not preceded by an RTS/CTS and are not acknowledged by their recipients. If an RTS does not result in a CTS packet in a specified time, a preset number of RTS retries are done before a link is declared as broken.

One of the most important features of 802.11 is the ad hoc mode, that allows users to build up Wireless LANs without an infrastructure (without an access point).

2.6 Network Simulator

Network simulator 2 (*ns-2*) [4] is the result of an on-going effort of research and development that is administered by researchers at Berkeley. It is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing and multicast protocols.

The simulator is written in C++ and a script language called Object Tool Command Language (OTcl). *ns-2* uses an OTcl script that defines the network (number of nodes, links), the traffic in the network (sources, destinations, type of traffic) and which protocols it will use. This script is then used by *ns* during the simulations. The result of the simulations is an output trace file that can be used to do data processing (calculate delay, throughput etc) and to visualize the simulation with a program called Network Animator (NAM). NAM is a very good visualization tool that visualizes the packets as they propagate through the network.

Mobility extension

A wireless mobility extension was developed by the Carnegie Mellon University [2] to ns-2. The extensions include detailed IEEE 802.11 wireless LAN and implementations of ad hoc routing protocols DSDV, AODV and DSR.

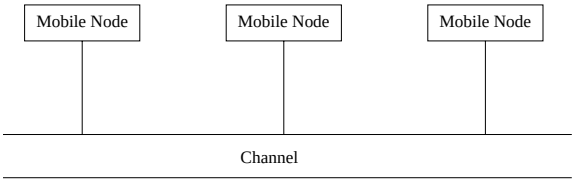

Figure 2.1: Shared media model.

Shared media The extension is based on a shared media model (like an Ethernet). This means that all mobile nodes have one or more network interfaces that are connected to a channel as in Figure 2.1. A channel represents a particular radio frequency with a particular modulation and coding scheme. Channels are orthogonal, meaning that packets sent on one channel do not interfere with the transmission of packets on another channel. The basic operation is as follows, every packet that is sent on the channel is received by all mobile nodes connected to the same channel. When a mobile node receives a packet, it determines if it is possible to receive the packet. This is determined by the radio propagation model, based on the transmitter range, the distance that the packet has traveled and the amount of bit errors.

Mobile Node Each mobile node, as shown in Figure 2.2, makes use of a routing agent for the purpose of calculation of routes to other nodes in the ad hoc network. Packets are sent from the application layer and are received by the routing agent. The agent decides a path that the packet msut travel in order to reach its destination and stamps it with this information. It then sends the packet down to the link layer. The link layer uses an Address Resolution Protocol (ARP) to decide the hardware addresses of neighboring nodes and map IP addresses to their correct interfaces. When this information is known, the packet is sent down to the interface queue and awaits a signal from the MAC protocol. When the MAC layer finds the channel free, it fetches the packet from the queue and hands it over to the network interface which in turn sends the packet onto the radio channel. This packet is copied and is delivered to all network interfaces at the time at which the first bit of the packet would begin arriving at the interface in a physical system.

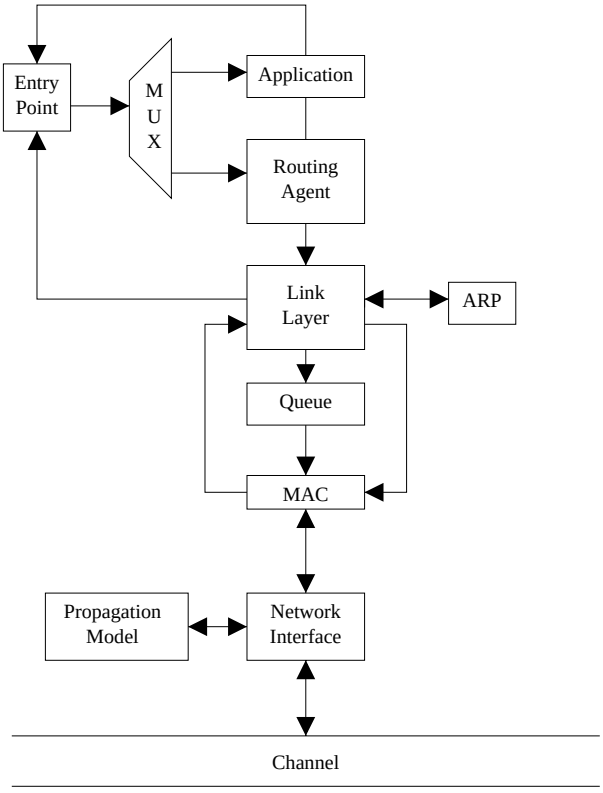

Figure 2.2: A mobile node.

The propagation model uses the transmit and receive stamps to determine the power with which the interface will receive the packet. The receiving network interfaces then use their properties to determine if they actually successfully received the packet, and sends it to the MAC layer if appropriate. If the MAC layer receives the packet free of any errors or collisions, it passes the packet to the node's entry point. From there it reaches a demultiplexer, that decides if the packet should be forwarded again, or if it has reached its destination node. If the destination node is reached, the packet is sent to a port demultiplexer, that decides to what application the packet should be delivered. If the packet should be forwarded again the routing agent will be called and the procedure will be repeated.

Chapter 3

Low-overhead Proactive Routing Techniques

In this chapter, we describe a new routing protocol for MANETs, called Adaptive Distance Vector (ADV), which uses load and mobility-based criteria to make the Distance Vector protocol dynamically adapt to node mobility and network load. While ADV uses routing updates to disseminate routing information, its routing overhead varies with load and mobility. Thus ADV exhibits some characteristics of on-demand protocols. Later in the chapter we describe a proactive technique in the on-demand AODV, aimed at reducing packet latencies in high mobility networks.

3.1 Adaptive Distance Vector - ADV

The Adaptive Distance Vector (ADV) starts with a basic distance vector algorithm that uses sequence numbers to avoid long-lived loops [20, 27]. ADV uses routing updates to learn and maintain routes just like any distance vector algorithm. However, we reduce the routing overhead propagated by varying the size and frequency of routing updates in response to traffic and node mobility. First, we maintain routes to only active receivers to reduce the number of entries advertised. Secondly, we adaptively trigger partial and full updates such that periodic full updates (used in RIP, IGRP etc. with 30-90 second periods) are obviated. We describe below how these effects are achieved.

3.1.1 Varying the number of active routes maintained

To reduce the size of the routing updates, ADV advertises the routes for active receivers only, unlike in the previous DV protocols wherein the updates propagate routes for all the nodes in the network. A node is an *active* receiver if it is the receiver of any currently active connection. A routing table entry is tagged with a special flag, called *receiver flag*, to indicate if the destination is an active receiver.

At the beginning of any new connection the source broadcasts an *init-connection* control packet in the network advertising that its destination node is an active receiver. All the nodes will turn on the corresponding *receiver flag* in their routing tables and start advertising the routes to the receiver in future updates. The target destination node upon receiving the init-connection packet responds, if it is not an active receiver already, by broadcasting a *receiver-alert* packet with its present sequence number. With a pair of broadcasts, all the nodes will know the active receiver and routes to it quickly.

When a connection is to be closed, the source broadcasts an *end-connection* control packet throughout the network indicating that the connection has been terminated. If the destination node has no additional active connections, then it broadcasts a *non-receiver-alert* control packet throughout the network that it has ceased to be an active receiver from now on. Then the nodes turn off the corresponding *receiver flag* in their routing tables and routes to this node are not advertised in future updates. Thus, the connection-initiation and connection-termination processes will help in varying the number of active routes in the network dynamically with the number of active connections open.

Since the first hop of an *init-connection* packet should be free of collisions, it is preferable to use some sort of reliable 1-hop broadcast mechanism [15]. Even if the packet is lost in the 1-hop broadcast, the source will advertise the receiver's entry with its receiver-flag set (metric may be set to infinity) in all future updates. This method of advertising an active receiver would, of course, be slower than the connection initiation process. The connection-initiation process in ADV differs from the route discovery process in the on-demand algorithms such as AODV and DSR, since in ADV each source sends only one init-connection packet at the beginning of a new connection. After that usual routing updates are used to maintain routes to the destination. Also the main purpose of the route discovery process in the

on-demand protocols is to get a route to the destination in case one is not available. But the connection-initiation process in ADV is mainly intended to advertise a destination as an active receiver, though as a side effect the routes to the destination are also known to all the nodes initially.

3.1.2 Varying the frequency of routing updates adaptive to network conditions

We now describe how the frequency of the routing updates can be changed with load and the mobility of the network. To keep track of load and mobility conditions, we maintain several variables. Some of the variables are defined for each routing entry in the routing table while the others are defined at the node-level. Later we explain how these variables are combined to determine when a routing update is broadcasted.

Variables used in each routing entry

The important variables that are part of each routing entry in the routing table are described in detail below. These are in addition to the usual fields used in DV algorithms (Table 2.1). The additional fields are also shown in Table 3.1 for easier reference.

Routing table entry	**Function**
Packets queued	Number of data packets waiting for a route to this destination
Packets handled	Indicates if this node is a forwarding node for this destination
Degree of propagation	Indicates the freshness of the route in the entry
Receiver flag	Indicates whether this node is an active receiver or not
Advertisement flag	Indicates whether this entry should be advertised in the next update or not

Table 3.1: Additional fields in a routing table entry at a node in ADV.

Packets queued: This variable counts the number of data packets buffered at a node waiting for a route to the destination corresponding to this routing entry. This count is incremented whenever a data packet is buffered for want of a route and is decremented whenever a data packet is freed from the buffers after obtaining a route to the destination. A non-zero count indicates the need for a fresh valid route for this destination immediately.

Packets handled: This variable indicates whether the node, having this routing entry, is a forwarding node to a destination for any of the node's neighbors. A source is also considered as a forwarding node as both need to maintain valid routes to the destination always. This count is incremented every time a packet is forwarded for a particular destination and is halved whenever an update containing an entry to this destination is propagated. However this variable becomes redundant if the packets queued field has a non-zero count.

Degree of propagation: This variable indicates the freshness of a routing entry. The freshness is determined in the form of new valid/invalid entry with a higher sequence number. It is set to 1 if the routing entry possesses, or has been updated with, a higher sequence number, otherwise set to 0. A value of 1 is used to indicate, in a future update containing this entry, that the receivers of the update should propagate this entry further to their neighbors.

Receiver flag: This flag is used to indicate if the destination is an active receiver or not (see Section 3.1.1). The routing entry is included in the updates only if the flag is set.

Advertisement flag: This flag determines if this routing entry is to be propagated in the next update or not. The flag is set if the routing entry has any fresh valid route with either a higher sequence number or a better hop count or has any invalidation to be propagated. The detailed conditions under which the flag is set is given in the Table 3.3. Unlike the *degree of propagation*, this flag will not help trigger an update in the neighbor node. (A node with its degree of propagation set to 1 will also have the advertisement flag set.)

Node-level variables

Only one copy of each of the following variables are maintained in each node.

Trigger meter: A node should trigger an update if it has a considerable number of packets waiting in buffers for routes to various destinations, or if one or more neighbors make a request for fresh routes.

Also a node has to propagate, at the earliest, any fresh valid/invalid route received with a higher sequence number. However, instead of triggering an update immediately after encountering any of the above conditions, if a node waits until it sees a sufficient need to trigger an update, the routing overhead could be enormously reduced. One way of implementing this method is to first quantify all the events, which have a capability to trigger an update, that have taken place since the last update. Then a check is made if the node has crossed a critical value, which would ensure the propagation of a routing update. Therefore, to keep track of all the events that force a routing update, we maintain a special variable called *trigger meter*.

The trigger meter is associated with the constants TRGMETER_FULL, TRGMETER_HIGH, TRGMETER_MED and TRGMETER_LOW, given in descending order of their values. The trigger meter is incremented by an appropriate level depending on the priority with which the node should trigger an update, upon receiving an update. After processing an entire update, if the trigger meter is modified then a check is made if the trigger meter exceeds the value of TRGMETER_FULL. If so a full update is immediately scheduled. If the trigger meter crosses a threshold value (explained below) but less than TRGMETER_FULL, then a partial update is immediately scheduled. Otherwise, no updates are scheduled.

Trigger threshold: The trigger threshold is used to decide when a partial update needs to be triggered. After processing an update, if it is not time to advertise a full update, a node checks if the trigger meter has crossed this threshold value. If so, a triggered update is immediately scheduled for transmission. The trigger meter is reset to zero after scheduling any update. This trigger threshold is changed dynamically based on the recent history of trigger meter values at the time of previous triggered updates. The computation of this threshold value is explained later in the section.

To reduce thrashing due to updates, we ensure that at least 500 ms elapse between any two updates triggered by a node. Even if an update is scheduled within 500 ms of triggering a previous update, it is delayed till the minimum elapsing time has expired. Also, at the time of triggering an update, a node

checks if it has any previous update still waiting in the interface queues pending transmission. If there is an update pending, then it is modified to reflect the new changes also; thus, avoiding an additional update.

Since full updates are triggered when the trigger meter value is high enough, we obviate the need for a periodic full update. However the mechanism to transmit periodic full updates still exists in case there is a need for them, just as in other distance vector algorithms.

Neighbor changes: In order to capture the network mobility, we keep a count on the number of neighbor changes taking place in a period of fixed number of full updates (say 5). The neighbor changes for a node include the nodes coming into the 1-hop neighborhood and those going out of its 1-hop neighborhood. The number of nodes going out of the 1-hop range can be determined by the number of link breakages whereas those coming into the range can be determined when an update is received from a neighbor whose metric is > 1 previously. If the number of neighbor changes exceeds a preset number, then we categorize the mobility as HIGH_SPEED or else as LOW_SPEED network.

Buffer threshold: When a new packet is buffered for lack of a route, a check is made if the number of packets already buffered exceeds a preset number, called buffer threshold, or not. If it exceeds it is considered to be high time a node trigger an update to get some fresh routes. Therefore, in such cases, the trigger meter is incremented by TRGMETER_MED in oder to gradually force an update.

Sending routing updates

This section explains the process of sending routing updates at a node. First we give the structure of a routing update entry and then we explain in detail the processing of a routing update at a node.

A node identifies an active receiver at any given moment, if its *receiver flag* is set in the corresponding routing table entry at that time. In a full update, a node includes all the entries of the active receivers, even if there is no advertised need for any of such entries. In a partial update, a node includes all entries of the active receivers whose *advertisement flag* has been set. The conditions under which

Destination IP address (32 bits)				
Next hop IP address (32)				
Sequence number(16)	Metric(8)	Is_receiver(1)	Expected_response(2)	Unused(5)

Table 3.2: Fields in a routing update entry. Total entry size is 12 bytes. The sequence number is the latest sequence number of the destination known. Metric indicates the hop count to the destination. Is_receiver flag indicates if the destination is an active receiver. Expected_response gives the priority with which a node wants to have a fresh valid route or to propagate this valid/invalid route further.

the advertisement flag of a routing entry is set are given in Table 3.3. Sequence numbers are used in a manner similar to their usage in DSDV except that every update (full and partial) results in a new higher sequence number.

With every routing update entry, a node sends an expected response value of ZERO (bit sequence 00), LOW (01), MEDIUM (10) or HIGH (11). The expected response values are determined using the following rules.

- In a HIGH_SPEED network, an expected response of HIGH is given when there are packets waiting for this route in the buffers (Queued $>$ 0) or the degree of propagation has been set to HIGH (see Table 3.3).
- In a HIGH_SPEED network, an expected response of MEDIUM is given when the node is a forwarding node to the routing entry's destination for any of its neighbors (Packets handled $>$ 0).
- In a LOW_SPEED network, an expected response of LOW is given when there are packets waiting for this route in the buffers (Queued $>$ 0) or the degree of propagation has been set to HIGH (see Table 3.3).
- If none of the above criteria apply then the expected response is set to ZERO.

The expected response value in each update entry essentially determines the priority with which a node, receiving this update, should respond to this advertised need for a fresh route. It also determines the need to further propagate a (valid/invalid) route, the freshness of which is indicated by a HIGH value for the degree of propagation.

Processing received updates

The conditions under which a node updates its routing table upon receiving an update are specified in detail in Table 3.3. However in all the cases, a node copies the Is_receiver flag value from the update, if the corresponding destination sequence number received is greater than or equal to the one present in the node. This helps in identifying the active receivers among the nodes dynamically, even if the *receiver-alert* and *non-receiver-alert* control packets are lost.

My_Entry	Received_Entry	Condition	Action
Valid	Valid	my_seqno < recv_seqno	Update My_Entry with Received_Entry Set the Advertisement flag For a forwarding node, degree of propagation = 1
		my_seqno == recv_seqno && my_metric < recv_metric	Update My_Entry with Received_Entry. Set the Advertisement flag
		my_seqno == recv_seqno && my_metric == recv_metric	Do not advertise this route any more
		my_seqno > recv_seqno || (my_seqno == recv_seqno && my_metric < recv_metric)	Set the Advertisement flag
Valid	Invalid	my_seqno < recv_seqno && my_hop = source of this update (see DSDV, Page 11)	Invalidate our entry since we are dependent on this neighbor Set the Advertisement flag For a forwarding node, degree of propagation = 1
		my_seqno > recv_seqno	Set the Advertisement flag Degree of propagation = 1
		recv_dest == My_Address && my_receiver_flag == TRUE	Trigger meter += TRGMETER_FULL Degree of propagation = 1
Invalid	Valid	my_seqno < recv_seqno	Update My_Entry with Received_Entry Set the Advertisement flag For a forwarding node, degree of propagation = 1
		my_seqno > recv_seqno	Do nothing
Invalid	Invalid	my_seqno < recv_seqno	Copy the recv_seqno into our entry
		my_seqno > recv_seqno	Do nothing

Table 3.3: Processing a routing update entry. my_seqno, my_metric, my_hop indicate the destination sequence number, hop count and the next hop node in the node's routing table. recv_seqno, recv_metric, my_hop indicate the values for the respective fields in the received routing update entry. The trigger meter manipulation is explained in detail below.

After each entry in the received routing update is processed, the trigger meter is incremented by an appropriate constant that is proportional to the expected response value. The trigger meter is incremented by TRGMETER_HIGH, TRGMETER_MED or TRGMETER_LOW for an expected response of

HIGH, MEDIUM or LOW respectively. The trigger meter remains unchanged for an expected response of ZERO.

In addition to the above cases, the trigger meter is incremented by TRGMETER_FULL when the receiving node, being an active receiver, finds out that its routing entry is invalid in the update received from a neighbor. This would certainly trigger a full update immediately after processing the update. The intention in doing this is to keep the all the one-hop neighbors informed of the presence of an active receiver within their neighborhood; so that, they can act as forwarding nodes to that receiver.

After processing all the update entries, the accumulated trigger meter value is checked to see if it exceeds TRGMETER_FULL in which case a full update is immediately scheduled for transmission. If not, the trigger meter is checked if it crosses the trigger threshold in which case a triggered update is immediately scheduled for transmission.

Processing data packets

In processing a data packet, that either originated in the same node or being forwarded by the node, each node checks the routing table for a valid route to the intended destination. If such an entry exists, then the data packet is immediately forwarded to the next hop node that is specified in the routing entry. Otherwise, the data packet is buffered until a valid route is available. Non-availability of routes suggests that the frequency of routing updates should be increased in order to maintain routes more up to date. Therefore, while buffering a new packet, if the number of packets buffered exceeds the buffer threshold then the trigger meter is incremented by TRGMETER_MED in order to gradually force an update.

Trigger threshold computation

The trigger threshold value for a node, that is either an active receiver or has packets buffered beyond the buffer threshold, is changed dynamically based on the recent history of trigger meter values at the time of previous triggered updates. The trigger threshold value is initially set to TRGMETER_HIGH. Each such node keeps track of number of triggered updates it has done, the sum of trigger meter values

at the time of each trigger, and the duration since the last full update. The trigger threshold value is computed every time it performs a full update using the following rules:

- The average trigger counter value per triggered update is computed by dividing the sum of trigger counter values (accumulated in a special variable since trigger meter is reset after every update) with number of triggered updates since the last full update (if one or more triggered updates are done). If no triggered updates are done by this node since last full update, the average is set to a high value of TRGMETER_HIGH. Let this average be t_n.

- The historical average trigger counter value for this node is t_h. The new historical average is computed as $t_h = (t_h + t_n)/2$. This is similar to the smoothing function, ($\alpha t_{old} + \beta t_{new}$) used in, for example, the computation of sample round trip time in TCP. Here we give equal weights of 0.5 to α and β in order to adapt to the mobility changes in the network rather quickly.

- Though the above conditions work well, it has been observed that some nodes, among active receivers or nodes with buffered packets, need to engage in more routing activity. To enable this, we check if the number of triggered updates done are less than the maximum expected number since the last full update, in which case the trigger threshold is set to a fraction of t_h (in order to increase the frequency of the triggered updates). The maximum expected number of triggered updates are calculated based on the minimum time between two triggered updates. Otherwise, it is set at t_h.

For the non-receiver nodes, that do not have packets buffered beyond the buffer threshold, the trigger threshold is set to a constant value that is commensurate with the speed of the network. For low speed networks, the trigger threshold is set to TRGMETER_HIGH and for high speed networks it is set to TRGMETER_MED.

The idea in computing the trigger threshold differently for different nodes, is to let the active receiver nodes and those that have too many buffered packets engage in routing activity that is adaptive to the

network conditions and at the same time discourage some of the non-receiver nodes from transmitting more than necessary updates.

3.1.3 Dual nature of routing updates in ADV

In all the previous distance vector protocols like the RIP [20], whenever a node changes the metric for a route in the routing table, it is required to send a triggered update almost immediately even if it is not time for the next regular update message. The triggered updates thus propagated can cause excessive loads on networks with limited capacity or with many number of nodes. To avoid this overhead, the triggered updates may be delayed by a small random time between 1 and 5 seconds during which, if other route changes occur they will all be consolidated into one triggered update. But such delaying is beneficial only in static and low speed networks. Therefore in ADV, instead of a random wait time for any additional route changes, any advertised need for fresh routes may trigger an update. Each of the update entries received has an expected response value, which indicates the necessity of triggering an update. The expected responses themselves are dependent on the mobility and load conditions of the network. Unlike in other DV algorithms, packets waiting for routes is a sufficient reason for triggering an update. Also in other DV based protocols, periodic full updates are necessary to maintain fresh routes, whereas in ADV, one could set the periodic update interval to ∞ and still have full updates if conditions warrant.

In on-demand protocols, the need for a fresh valid route to an active receiver will immediately result in a route discovery process and the intermediate nodes also rebroadcasts the request immediately if the route to the receiver is unavailable. Also as the route replies are unicasted, they reach the intended source nodes reliably. However in ADV, a fresh valid route can only obtained from neighbor updates. So obtaining a valid route could take long time in ADV.

3.2 A Proactive Technique To Reduce Latencies in AODV

The Ad hoc On-demand Distance Vector (AODV) protocol which has been designed as an on-demand routing protocol has very high latencies, over 100 ms on average [7], especially at low and moderate network traffic in high mobility scenarios. In order to reduce the latencies in these situations, we have introduced a proactive technique into the AODV algorithm. In this technique, the receivers periodically (for example, 1 second intervals) send beacons that are broadcasted through out the network. A beacon entry has the receiver's sequence number, IP address, broadcast ID and the hop count. The hop count is incremented at each of the forwarding nodes. If the received sequence number (or a better hop count with the same sequence number) is higher than the one in the routing table, then the routing table is updated with the next hop set to the node that sent the beacon packet. If a beacon entry does not result in adding/updating the corresponding entry in the node's routing table, then the node does not propagate it further. Thus beacons refresh the routing entries for receivers in other nodes' routing tables, even before they are expired. Consequently the data packets will have fresher routes to its destinations without the need for a route discovery process. This, in turn, reduces packet latencies. However, this technique increases the routing overhead because of the additional beacon packets.

To reduce the number of beacons propagated, two almost similar methods were tried. In both the methods, the nodes accumulate all the beacons for a preset duration (1 second) and send consolidated beacons piggybacked on any outgoing route request (RREQ) packets, since both are of broadcast type. In one of the methods, called BCAODV, the consolidated beacons are piggybacked on RREQ only if there is a RREQ going out within 0.75 to 1.0 of the beacon interval, otherwise the consolidated beacon entries are sent as a separate beacon packet. In the other method, called BCPIGGY, the consolidated beacons are piggybacked on RREQs all the times. Clearly BCPIGGY does not add anymore routing packets to the network but transmits the same amount of extra beacon overhead bytes as BCAODV. However BCPIGGY will not reduce the packet latencies much, when compared to BCAODV, since the beacon entries may be delayed beyond its 1 second interval at a node waiting for RREQ packets.

Using simulations, we have found BCAODV to perform better than BCPIGGY. BCAODV reduces latencies by as much as 50% whereas BCPIGGY reduces only by 30%. Also BCAODV actually reduces the number of routing packets transmitted as it delivers the beacon entries without any delay; thereby, limiting the route discovery processes done. Therefore we have chosen BCAODV to represent the beacon technique when compared against AODV.

Chapter 4

Performance Analysis

We have compared the performance of ADV with AODV and DSR using extensive simulations. We have used AODV and DSR for comparisons since they have been extensively analyzed in literature [1, 7, 11, 14]. First we explain the various models and parameters used for node mobility and traffic patterns, followed by the metrics used for performance evaluations. Next we present the performance results.

4.1 Simulation Setup

4.1.1 Movement Models

Two models of network fields have been used for simulations. The first model has 50 nodes randomly placed on a square field of 1000m x 1000m at the beginning of a simulation. Nodes move continually with randomly chosen direction and speed. A node that reaches the boundary of a field exits the field and reenters it immediately from the other (parallel) side of the field with the same velocity. Such a network simulates nodes leaving and entering over a period of time, yet maintains a constant number of nodes in the simulation to facilitate performance comparisons. The node density is less compared to network fields used in other studies, yet has very few network partitions. This mobility pattern is based on the random Gaussian Markov model proposed in literature. The second network field consists of 100 nodes placed on a 2200m x 600m rectangular field in which nodes that reach edges bounce back into the

field. This mobility pattern is used in several other studies with varying field sizes and number of nodes [1, 7]. For this mobility pattern, the performance is dictated by the contention for the shared channel among the nodes in the middle of the field. All simulation results for the 100 nodes rectangular field are presented in Appendix A.

Each node travels a preset distance of 10 meters at a time. The direction and speed are chosen randomly. For a high speed network, the speed is uniformly chosen between 0 m/s and 20 m/s (72 Km/hr), and for a low speed network the speed is uniformly chosen between 0 m/s and 2 m/s. After traversing a preset distance, a node may pause for a few seconds before moving again. The following varying pause times have been used for simulations: 0, 30, 60, 120 seconds. A pause time of 0 seconds corresponds to continuous motion, and a pause time equal to the length of the simulation corresponds to no motion.

Since the performance of the routing protocols are very sensitive to movement patterns, 5 different scenarios were generated (with different random number seeds) for each pattern and each simulation point is averaged over these five scenarios. This way any arbitrary randomness is minimized.

4.1.2 Traffic Model

We have chosen our traffic sources to be constant bit rate (CBR) sources with 20, 40 and 60 connections. With 20 connections, only a part of the network is used. With 40 and 60 connections, almost all of the network is used. All the connections are peer-to-peer, and are started within 5 seconds of start of the simulation. The packet sizes are fixed at 64 bytes for all the simulations. In most simulations with no pause times, packet rates are varied from 0.25-10 packets/s to see performance of algorithms under varying traffic loads.

In simulations with constant number of connections, a warm-up of 300 seconds is used. Statistics are gathered for 500 seconds after the warm-up period. So these simulations capture mainly the steady-state behavior of a routing algorithm. In simulations with varying number of connections, no warm-up time is used. Such simulations capture transient-state behavior of the algorithms.

In simulations with varying loads, one or two algorithms may saturate early and could not be simulated at higher loads that other algorithms were able to sustain. For such simulations, the plots may look incomplete, though each algorithm is simulated until it saturated.

4.1.3 Metrics and Parameters

In comparing the protocols, the following metrics are used to evaluate them:

- *Average packet latency*: The time, in milliseconds, it takes for a data packet to reach its destination from the time it is generated at the source and includes all the queuing and protocol processing delays in addition to propagation and transmission delays.
- *Routing Overhead (pkts/s)*: The total number of routing packets transmitted at the routing layer per second during the simulation. For packets sent over multiple hops, *each* transmission of the packet (each hop) counts as one transmission. This is the routing overhead seen at the routing layer level.
- *Routing Overhead (Kb/s)*: The total number of bits transmitted per second, as routing packets and IP packet headers in the data packets. For DSR, this includes the source routing information in the data packets. This is the routing overhead seen at the routing layer level.
- *Throughput*: The total number of data bits delivered at the destinations measured in kb/s.
- *Packet delivery rate*: The ratio of the number of packets received by the CBR sink at the final destinations to the number of packets originated by the “application layer” CBR sources.
- *MAC overhead (pkts/s)*: The total number of routing packets transmitted at the MAC layer per second during the simulation. This includes the RTS, CTS, and ACK packets as well as the IP layer routing packets. This overhead seen at the MAC layer level gives a more complete picture of the actual number of routing packets being transmitted on the physical medium.

We have implemented the ADV on *ns-2*. The implementation exactly matches what has been described in Chapter 3. For DSDV and DSR, we have used the CMU's implementation. All the parameter values and optimizations used for DSDV and DSR are exactly as described by Broch et al. [1]. The AODV implementation has been provided by the AODV group. The implementation is the same as that used in [7].

Results from simulations with varying network loads are plotted with respect to offered (data) load in Kb/s. Results from varying pause times are plotted with respect to pause times. For clarity each plot is entitled with the number of CBR connections and maximum node speed. Some of the important parameter values used in various routing protocols are given in the separate tables below.

Parameter	Value
Periodic update interval	15 seconds
Minimum time between two triggered updates	1 seconds
Maximum packets buffered per node per destination	5
Periodic updates missed before link declared broken	3

Table 4.1: Values of various parameters used in the DSDV protocol.

Parameter	Value
Minimum time between two triggered updates	0.5 seconds
Maximum packets buffered per node	64
Buffer timeout	1 second
Buffer Threshold	5
TRGMETER_FULL	50
TRGMETER_HIGH	20
TRGMETER_MED	8
TRGMETER_LOW	5
Periodic update interval	∞

Table 4.2: Values of various parameters used in the ADV protocol.

Parameter	Value
Active route timeout	50 seconds
Request retries	3
Maximum packets buffered per node	64
Buffer timeout	30 seconds
TTL_START	1
TTL_INCREMENT	2
TTL_THRESHOLD	7

Table 4.3: Values of various parameters used in the AODV protocol.

Parameter	Value
Time between retransmitted requests	0.5 seconds
Maximum packets buffered per node	64
Buffer timeout	30 seconds
Primary route cache size	30 entries
Secondary route cache size	34 entries

Table 4.4: Values of various parameters used in the DSR protocol.

Link layer notification of broken links is used for ADV, AODV and DSR. This is not done in DSDV since [1] reports that the link-layer feedback increases routing updates transmitted. Therefore in DSDV, loss of a neighbor is detected when 3 consecutive periodic updates are missed from that neighbor.

4.2 ADV vs. DSDV

Figures 4.1 and 4.2 give the performances of ADV and DSDV for varying network loads in the 50-node square field.

We notice in Figure 4.1 that DSDV saturates at an offered load of only 50 Kb/s, whereas ADV can sustain a load of up to 90 Kb/s. It can also be seen that ADV provides higher packet delivery rates for all the loads and thereby better throughput. Though packet latencies are higher for ADV at low to moderate loads relative to DSDV, ADV can maintain almost the same packet latencies even for higher loads until it saturates.

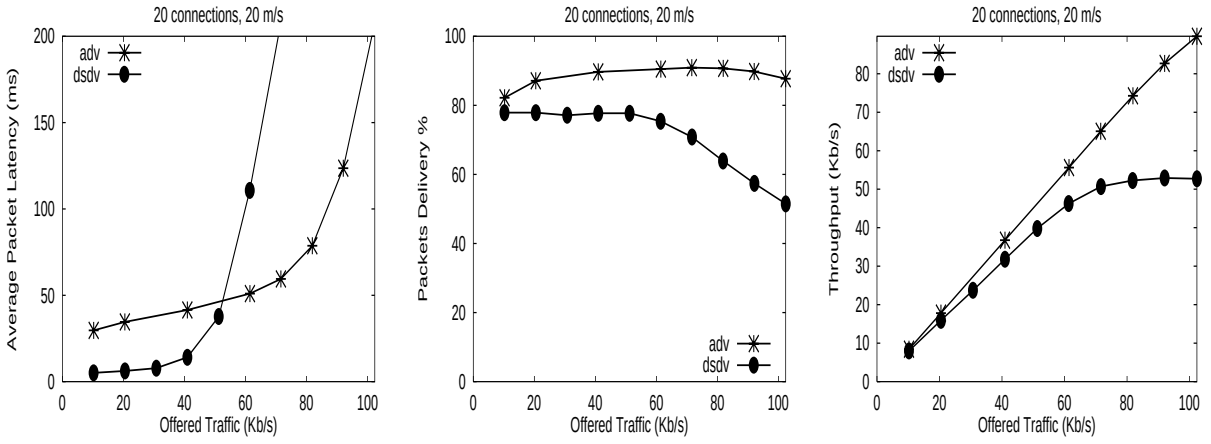

Figure 4.1: Data packet latencies, delivery rates and throughputs of ADV and DSDV for 20 CBR connections in the high speed 50 node square field.

In Figure 4.2, we observe that the routing overhead in Kb/s is lesser for ADV than DSDV since routing updates in ADV contain entries only for active receivers rather than for all the nodes in the network; thereby, resulting in smaller routing packets. But the number of routing packets per second are more in the case of ADV because the frequency of full updates increases, that is found to be about one full update every 3 seconds (at moderate loads), compared to one full update every 15 seconds in DSDV. We also observe that DSDV gives almost constant overhead transmitted; thereby, indicating the periodicity of the update entries rather than with any regard to the network load conditions. But in the case of ADV, the routing load increases with increasing offered load; thus, demonstrating its adaptive capabilities.

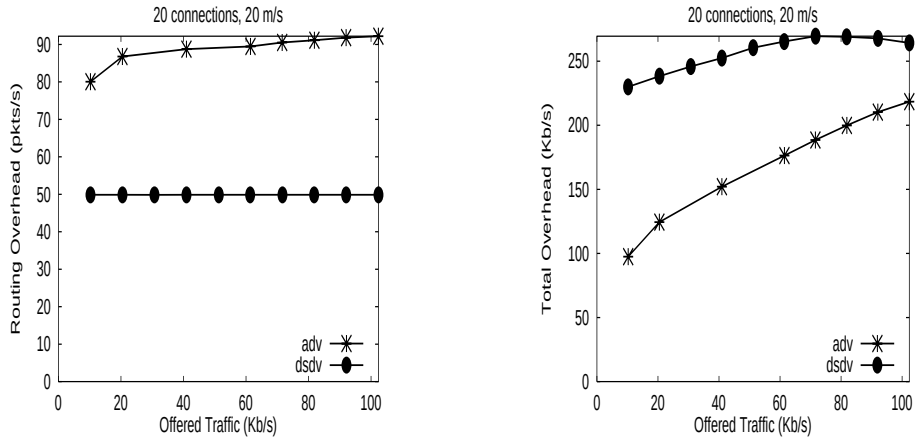

Figure 4.2: Overheads of ADV and DSDV for 20 CBR connections in the high speed 50 node square field.

On-demand protocols are preferred over DSDV because DSDV cannot handle high loads [1, 11, 14]. Also its packet delivery rates are lower than AODV and DSR. But with ADV performing significantly better than DSDV, especially in sustaining higher loads, giving higher packet delivery rates and lower packet latencies. It will be interesting to compare it with on-demand routing protocols like AODV and DSR. In the next section we compare the performances of ADV, AODV and DSR.

4.3 ADV vs. on-demand algorithms

4.3.1 Steady-state behavior at high speed

This section presents the performance comparison of various metrics among ADV, AODV and DSR protocols. The comparisons are done for 20, 40 and 60 CBR connections for the high mobility 50 node square field at steady-state conditions. As explained in section 4.1.2, steady-state behavior captures only the stable network conditions, after an initial warm-up time. All the protocols are run until they saturate. ADV sustains much higher traffic loads than AODV and DSR, which cannot be simulated at those high loads . This explains why some plots of AODV and DSR may look incomplete.

Packet latencies. We observe from Figure 4.3 that ADV gives the least average packet latenices at all loads compared to AODV and DSR. In fact ADV improves the latencies by more than 50% over AODV in all the cases and improves significantly over DSR especially for higher number of CBR connections.

ADV has lower packet latencies because of its proactive nature. It has routes to all the active receivers readily available at almost all the times. Whereas in AODV and DSR, routes are frequently found by route discovery mechanisms that incur a delay from the start of the route request propagation to the receipt of the first route reply. The snooping of the data packets for source routes in DSR, that reduces the need for route discovery processes, results in lower packet latencies in DSR compared to AODV.

We also observe in all the three cases, ADV gives consistently lower packet latencies till an offered load of 90 Kb/s. This indicates that ADV can sustain higher packet rates than the on-demand protocols. The on-demand protocols give very high latencies after a load of 40 Kb/s making them less attractive at higher loads.

Packet delivery rates. Figure 4.4 gives the packet delivery rates for all the three cases. We find that ADV maintains consistently high delivery rates for all the loads until it saturates. An interesting observation from Figure 4.4 is that ADV and DSR fail to give high delivery rates at very low loads (like

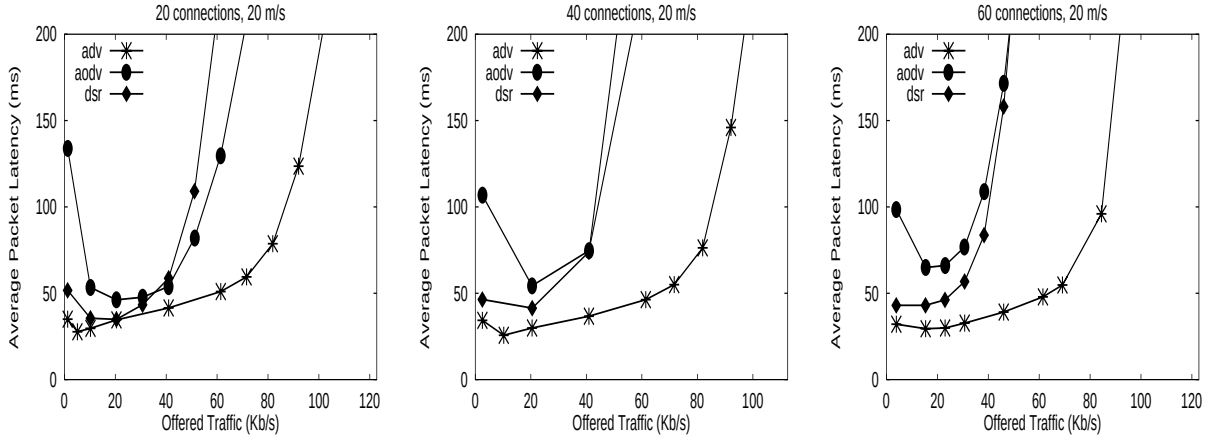

Figure 4.3: Data packet latencies of ADV, AODV and DSR for the high node mobility 50 node square field and various CBR connection cases.

1 packet for every 8 seconds). In ADV, a node is designated as a forwarding node if the number of data packets forwarded (Packets handled) are non-zero. The packets handled variable is incremented by 1 every time a packet is forwarded and is halved whenever an update with an entry corresponding to that destination is transmitted. Because of the less number of data packets forwarded at very low loads, the nodes in ADV fail to act as forwarding nodes since the packets handled variable is frequently made zero by the routing updates. In DSR, snooping is not effective (or even counter productive) at very low packet rates as it results in routes becoming stale. These stale routes if used might result in further dropping of data packets.

The three protocols give almost the same packet delivery rates until their respective saturation points.

Throughputs. Figure 4.5 gives the throughputs supported by the three protocols. The ADV protocol saturates at around 90 Kb/s in all the three cases; whereas, the on-demand protocols saturate very early at around 50-60 Kb/s. This shows the poor sustainability of the on-demand protocols at high loads. The low throughputs offered by the on-demand protocols at high loads is caused by the large number of routing packets (see Figure 4.6) that compete for the channel along with large number of data packets at high data rates, thus making the channel bandwidth scarce. This leads to both routing packets and data packets getting dropped from the interface queues. Also, as the routing packets find fewer chances to

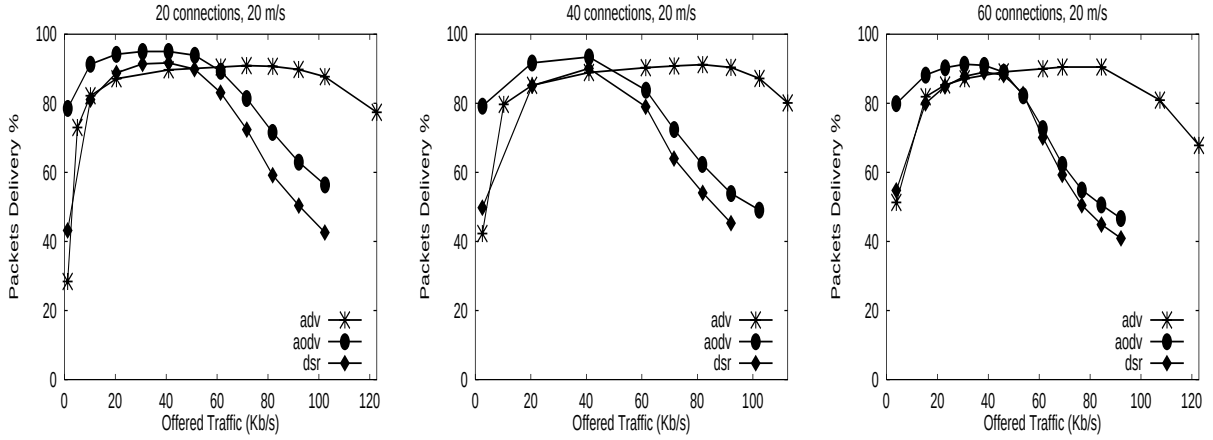

Figure 4.4: Data packet delivery rates of ADV, AODV and DSR for the high node mobility 50 node square field and various CBR connection cases.

propagate the route discovery processes take a longer time, thus dropping the data packets in the buffers at node. (The number of buffers maintained at a node are static which cannot cope with the large number of data packets at higher loads.)

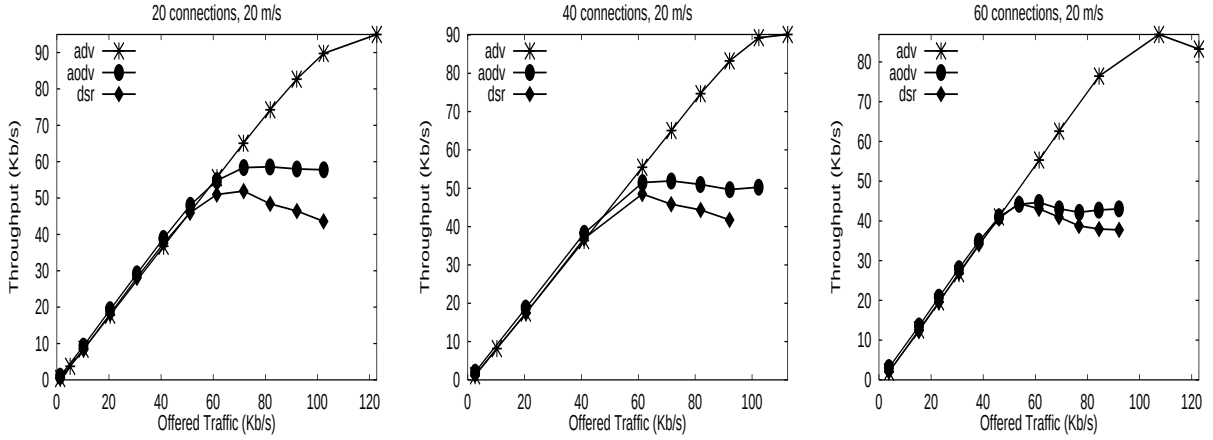

Figure 4.5: Data packet throughputs of ADV, AODV and DSR for the high node mobility 50 node square field and various CBR connection cases.

Routing overhead. Figure 4.6 gives the IP layer routing packets transmitted per second for the three protocols. ADV transmits the lowest number of routing packets per second and routing load remains nearly constant for all the loads. At moderate loads (30 Kb/s offered load) , ADV transmits 50% less packets than AODV for the 20 connections case and around 75% less packets for the 40 and 60 con-

nections cases. Even when compared to DSR, ADV transmits anywhere between 25-50% less routing packets.

An interesting observation in the case of DSR is that the proportion of unicast routing packets, i.e. route replies and route errors, constitute about 70% of total routing packets at low and moderate loads that increases to 80% at higher loads. The reason for this behavior is that, in DSR, the availability of routes is higher because of source route snooping and aggressive caching. As a result the route discovery processes result in less propagation of route requests and more number of nodes sending route replies. (DSR initially broadcasts the route requests only to the 1-hop neighbors and then to the entire network if no reply is received.) With more routes available most of the route discovery processes are completed within the 1-hop limit itself accounting for only one route request broadcast packet.) This large proportion of unicast routing protocols results in excessive MAC layer overhead (Figure 4.8) in terms of RTS, CTS and ACK exchanges; thus, clogging the wireless medium at higher loads.

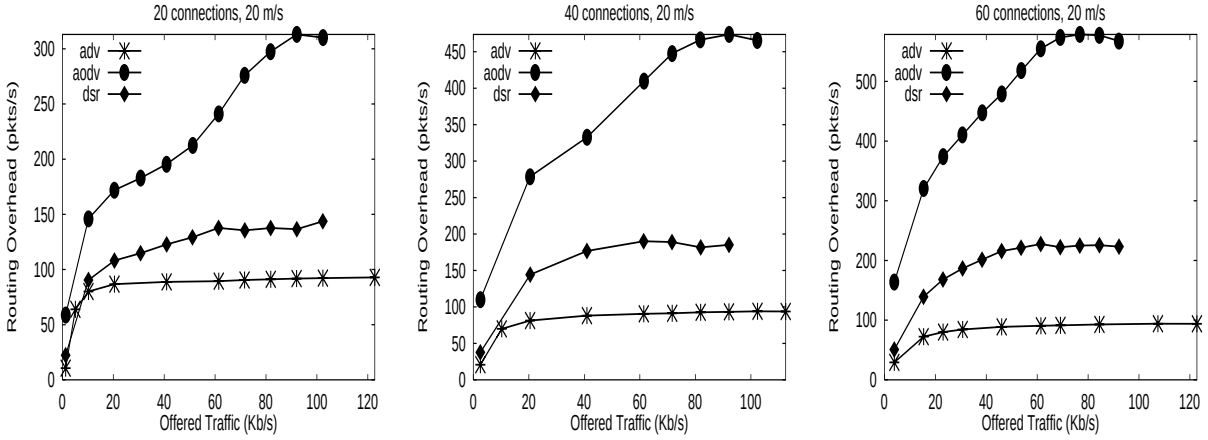

Figure 4.6: Routing overhead in packets/s of ADV, AODV and DSR for the high node mobility 50 node square field and various CBR connection cases.

We also see, from Figure 4.6, that DSR transmits fewer packets than AODV especially at moderate to high loads. As DSR uses caching and snooping techniques, its need for route discoveries is limited. But in the case of AODV, its route discoveries typically propagate throughout the network; thus, contributing more routing packets. Also DSR maintains multiple routes to the destinations in its cache. This redundancy would help in choosing the shortest route to the destination from among the routes.

The disadvantage in such redundancy is that the same shortest path is always chosen even if the route has become stale as no timers are maintained to convey the freshness of a route.

In ADV, as the offered load increases beyond 50 Kb/s the number of routing packets transmitted becomes constant at about 100 packets/second. Interestingly this corresponds to the maximum number of packets that all the nodes can transmit in one second after taking into account a 500 ms elapsing time between any two updates to prevent thrashing. Thus, even though the ADV is forced not to propagate routing updates beyond a limit, its performance is better than the other two protocols at higher loads. This is especially seen in higher packet delivery rates and higher sustainable throughputs.

Figure 4.7 gives the routing overhead transmitted in Kb/s. We can see that ADV transmits higher number of routing bytes than the other two on-demand algorithms because ADV tries to keep the routes fresh for all the active receivers. That is why we observe that the difference in overhead bytes compared to AODV and DSR increases as the number of connections (receivers) increase. However it should be noted that the cost to acquire the medium to transmit a packet is significantly more expensive in terms of power and network utilization than the incremental cost of adding a few bytes to an already existing packet. Hence ADV with its low packet overhead is a better protocol in terms of IP routing overhead transmitted.

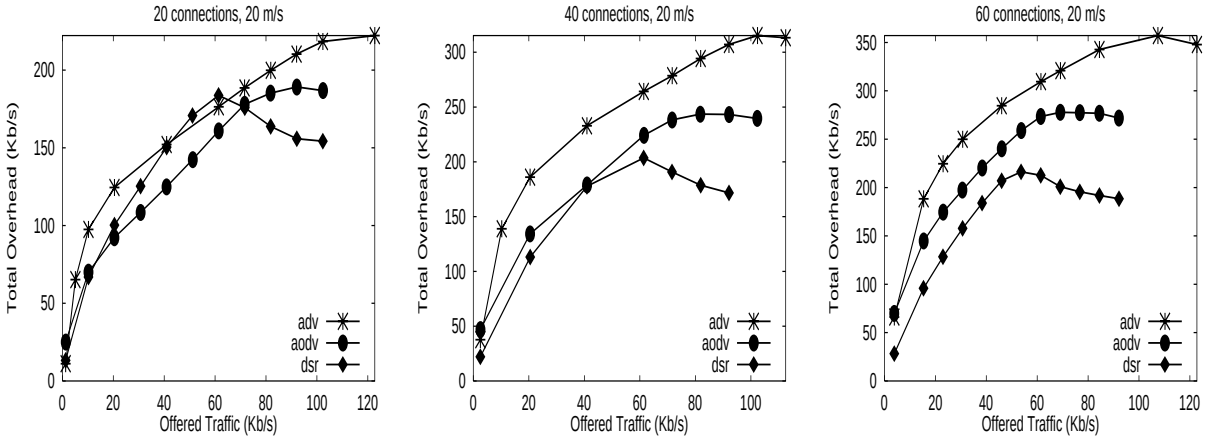

Figure 4.7: Routing overhead in Kb/s of ADV, AODV and DSR for the high node mobility 50 node square field and various CBR connection cases.

MAC layer level overhead in pkts/s. Figure 4.8 gives the MAC layer level routing overhead. This includes all the RTS, CTS, ACK exchange packets and all the IP layer routing packets. This metric gives us an indication as to how many routing packets are actually transmitted on the physical medium.

ADV transmits fewer routing packets at the MAC layer level than AODV and DSR because ADV has less number of IP layer routing packets (Figure 4.6) and, more importantly, no unicast routing packets among them. In AODV and DSR, the route reply and route error packets are unicast routing packets which incur additional MAC overhead in terms of RTS, CTS and ACK packets. Thus the total MAC layer level routing packets in AODV and DSR are more than that seen in ADV, which at moderate to high loads can be as large as 25% more packets.

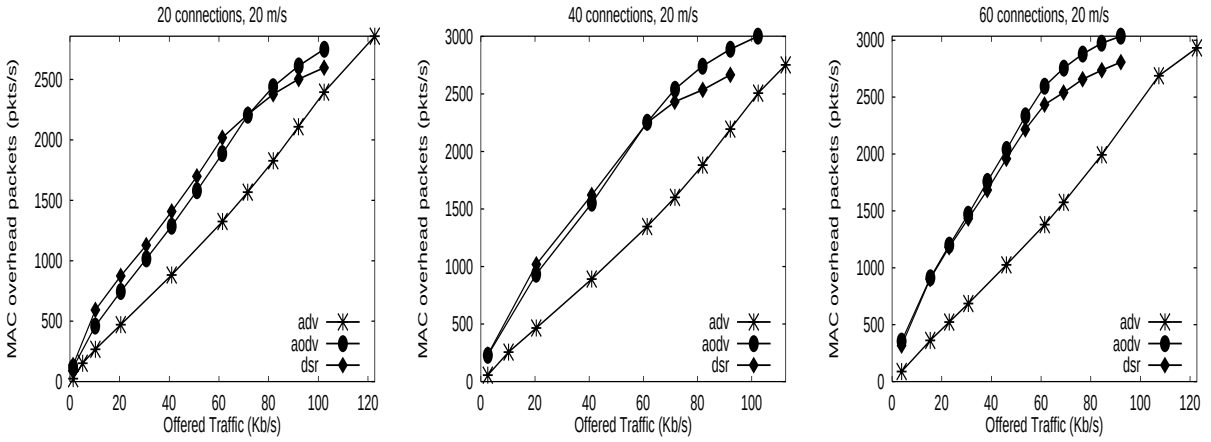

Figure 4.8: MAC layer overhead in pkts/s of ADV, AODV and DSR for the high node mobility 50 node square field and various CBR connection cases.

We have also observed that, out of the MAC layer level routing packets, the RTS, CTS and ACK packets far outnumber the number of IP layer routing packets. This is because of the large number of data packets all taking an average of around 3 hops to reach their corresponding destinations with each hop unicasted along the way. Thus the overhead seen at the MAC layer in terms of Kb/s gives almost the same performance differences between the three protocols since the size of the RTS, CTS and ACK packets are the same in all the cases. From the simulation data, we have also observed that the number of RTS packets are as much as twice the number of CTS packets at high loads and high mobility. This is because of the frequent RTS retransmissions for errors due to collisions.

For loads above 80 Kb/s, the increase in MAC layer level overhead in ADV is due to the larger number data packets transmitted at high loads; whereas, in AODV and DSR it is mainly due to the increase in routing overhead. One more interesting observation that has been made from the simulations is that ADV and DSR take lesser average number of hops (around 3.2) than AODV (around 3.6) to transmit a data packet.

Additional observations. The poor performance of the DSR at high loads is due not only to the large number of routing packets transmitted at the MAC layer but also because of the stale routes chosen to transmit the data packets. The data packets transmitted with such stale routes will consume additional network bandwidth even though they will eventually be dropped and will also corrupt the route caches in other nodes. The problem of stale routes can arise depending on the way a route is chosen from among the multiple cache entries. The only criteria applied is the route length, no matter how "old" the entry. Thus the problem of stale routes exist unless a timer is set to delete all the expired entries from the cache. Although the route errors remove some stale routes from the caches, it is still possible that snooping on the data packets using stale source routes, may still add more stale routes to the caches.

4.3.2 Steady-state behavior at low speed

This section presents the performance comparison of various metrics among ADV, AODV and DSR protocols for the low speed mobile networks. The comparisons had been done for 40 CBR connections for the low speed 50 node square field at steady-state conditions and shown in Figures 4.9 and 4.10. The results for 20 and 60 connections are almost similar and; hence, are not presented.

ADV gives lower latency and higher peak throughput than AODV and DSR. But the packet delivery rates of ADV are slightly lower than the other two protocols at all loads. Most of the packets dropped are due to the non availability of the routes to the destinations. This may be due to the triggering criteria failing to transmit enough number of updates, in a timely fashion, to keep the routes fresh.

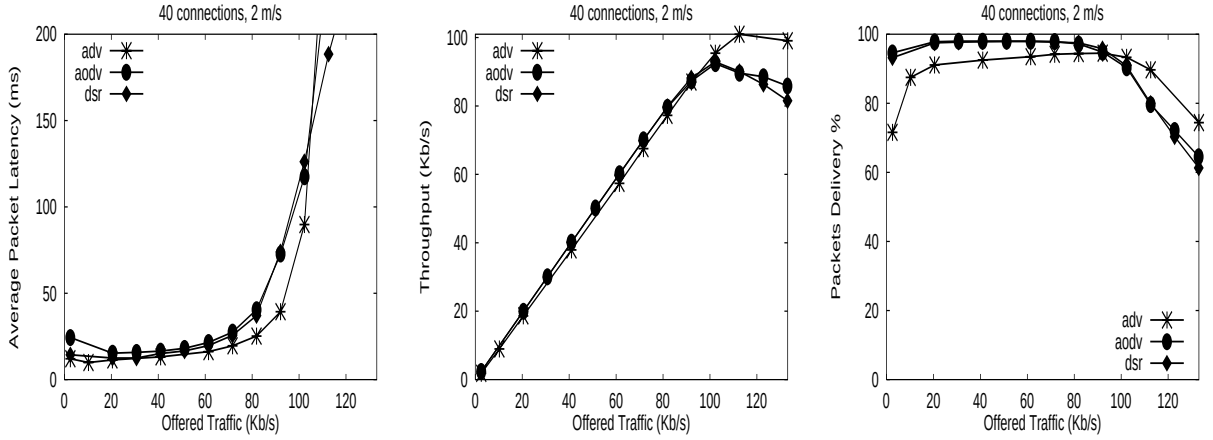

Figure 4.9: Data packet latencies, throughputs and delivery ratios of ADV, AODV and DSR for the low speed 50 node square field and 40 CBR connections.

From Figure 4.10, we observe that ADV has the highest overhead both in terms of packets/s and Kb/s. Interestingly, what makes the DSR transmit more routing overhead in Kb/s than AODV, unlike the high speed case, is that the source routing overhead becomes a significant factor contributing to the routing overhead. With increasing load, more number of data packets carry source routes and thus the overhead increases more “steeply” than AODV. Though ADV transmits lesser number of MAC layer level routing packets (up to 60 Kb/s load) it can be attributed to its lower delivery rate.

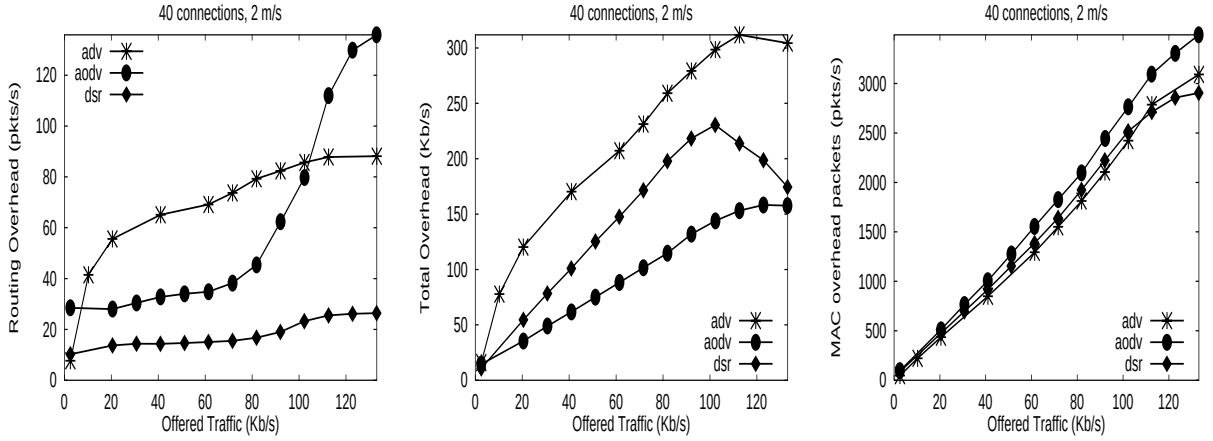

Figure 4.10: IP routing overhead in terms of packets/s and Kb/s and overhead packets/s as seen at the MAC layer of ADV, AODV and DSR for the low speed 50 node square field and 40 CBR connections.

The reason for ADV not giving higher packet delivery rates is due to the fact that the trigger meter is not incremented by an appropriate amount each time it is incremented. Because of this, the propagation of routing updates is delayed and thus data packets may be dropped for want of routes. Since an expected response value of LOW is assigned for routing entries under low speed conditions (see Section 3.1.2), the trigger meter is incremented only by TRGMETER_LOW. This would result in reduced number of updates triggered at a node. Therefore, at low speeds, the trigger meter should be incremented in larger proportions so as to respond more quickly to the advertised requests for fresh routes.

4.3.3 Adaptive behavior of ADV

Figures 4.7 and 4.10 show the adaptive behavior of ADV with respect to load. As the offered load increases, the routing overhead transmitted also increases which is representative of any on-demand protocol.

This section explains how the ADV is adapting to the network mobility conditions. For easier comparison we present the ADV performance at both the low speed and the high speed in the same graph. Figure 4.11 gives the performance of ADV for 40 connections in low speed and high speed 50-node square fields. They are named low-adv and high-adv respectively.

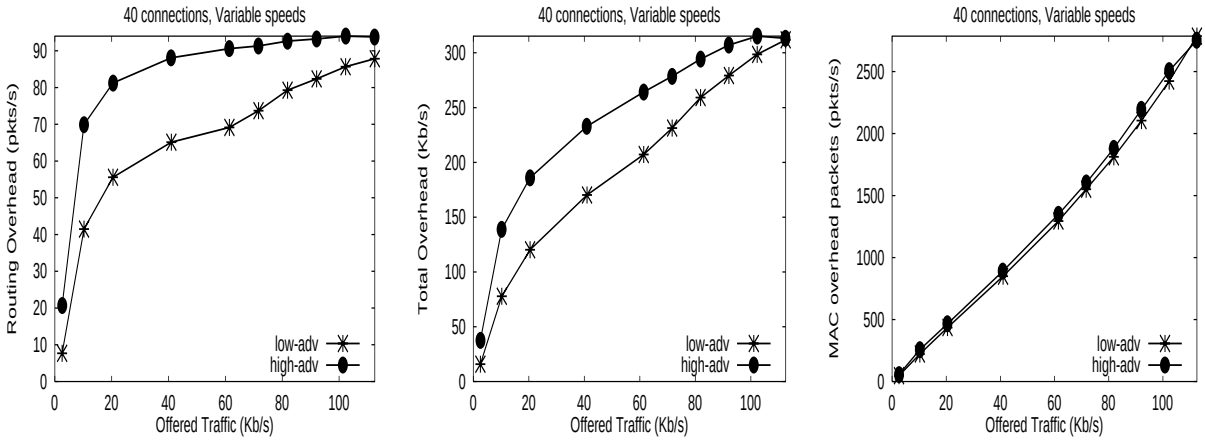

Figure 4.11: Routing overhead in terms of packets/s and Kb/s and overhead packets/s as seen at the MAC layer of ADV protocol at both low and high speed 50 node square field and 40 CBR connections. *low-adv* represents the ADV run on low speed network and *high-adv* represents the ADV run on high speed network.

Observing the IP layer routing overhead transmitted per second, we can see that the ADV transmits higher routing load for high speed than for low speed that conforms to the mobility-based criteria in the ADV. Because the number of route invalidations are more in a high speed network, ADV transmits more routing overhead with increase in speed in order to maintain fresh valid routes to all the active receivers.

The MAC layer level routing overhead is almost the same at both the speeds. The lower number of IP layer routing packets in the low speed case are becoming insignificant when mixed with the RTS, CTS and ACK packets.

4.3.4 Steady-state behavior at different pause times

This section describes the performance metrics for the protocols at different pause times for the 50-node square field. Only the 40 connection case is considered and the data packet rate is 1.5 packets/sec which offers a load of 30.72 Kb/s. Figures 4.12 and 4.13 show the characteristics of the three protocols at different pause times.

For all the cases, ADV has the lowest latency. ADV latencies remain nearly the same but AODV and DSR show a drastic decline in the latency values as the pause time increases. This seems to suggest that AODV and DSR are likely to exhibit significantly worse performance as nodes become more and more mobile (less pause times). But the throughput plot shows that ADV performs rather poorly at 30 seconds and 60 seconds pause times. The reason for this is the same as explained in Section 4.3.2, i.e., the slow triggering in the low speed networks results in the delayed propagation of updates. Again there is a need to fine tune our adaptive criteria used to trigger in the slow network environments at different pause times. (It should be noted that the nodes in a pause time environment wait for the pause time asynchronously. So when the network is considered at any moment, there will be network changes all the times, but they would be very less in number. Hence the pause time networks, even though the nodes move at high speeds, get categorized as low speed networks.)

Figure 4.13 shows that ADV has the lowest routing overhead in packets/s at all the pause times. What is interesting to observe is that DSR transmits the highest overhead in Kb/s especially at non-zero

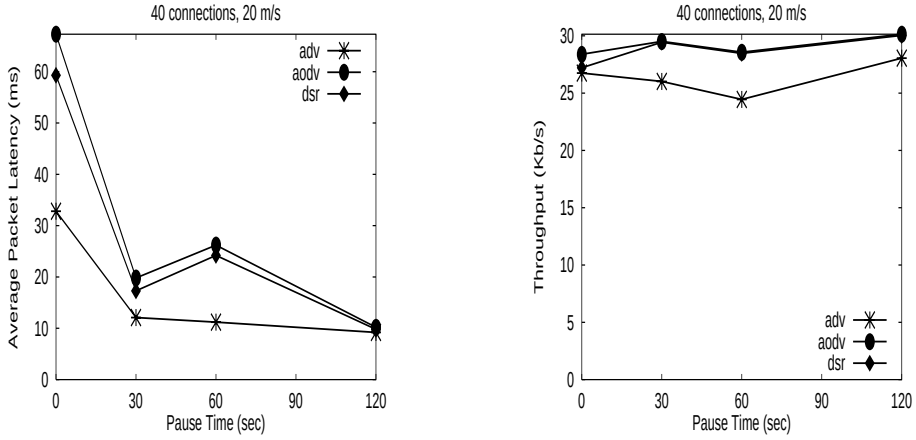

Figure 4.12: Average packet latencies, throughputs and delivery ratios of ADV, AODV and DSR at different pause times for the high node mobility 50 node square field and 40 CBR connections.

pause times. This can be attributed to the source routing overhead that is significantly higher at non-high speeds, as observed in Section 4.3.2.

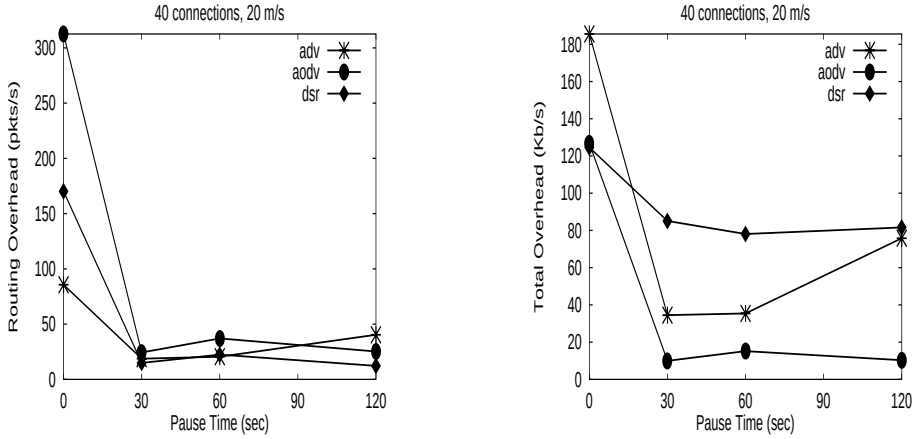

Figure 4.13: Routing overhead in packet/s and Kb/s of ADV, AODV and DSR at different pause times for the high node mobility 50 node square field and 40 CBR connections.

AODV 's routing overhead decreases dramatically with increase in pause times indicating that it is a better algorithm for low speeds or infrequent node movements. Also, when compared to DSR, AODV transmits fewer routing bytes.

This section again shows the adaptive behavior of the ADV with regard to the mobility conditions in a network. The basic character of any on-demand protocol is demonstrated in the shape of its overhead curve. The overheads in AODV and DSR drop as the mobility rate drops, as expected. ADV, in spite of its proactive approach, has almost a similar overhead curve as that of the other on-demand protocols. Thus we can say that ADV exhibits on-demand characteristics.

4.3.5 Transient state behavior

This section describes the transient state conditions of the network for the three protocols. The traffic is generated by starting 10 new CBR connections started every 60 seconds. This happens for five 60-second intervals at the end of which there will be altogether 50 connections in use. (All the 10 connections initiated in an interval start within the first second of that interval.) The network is simulated for two additional 60-second intervals during which no additional connections are initiated.

The load offered per connection is one 64-byte packet/second. When all the 50 connections are used, i.e. in 5th, 6th and 7th intervals starting at 240, 300 and 360 seconds respectively, the load offered will be 25.6 Kb/s which is comparatively an average load for the three protocols.

High speed network

Figure 4.14, gives the packet latencies, packet delivery rates and throughputs in transient state conditions of all the three algorithms in the high speed case. ADV offers the least latencies which is 70% less than AODV and 50% less than the latency offered by DSR. Packet delivery rates and throughputs seem to be almost the same for AODV and ADV with DSR performing slightly worse.

An interesting observation in Figure 4.14 is that packet latencies for AODV and DSR increase after 300sec, though no new connections are initiated. This seem to indicate that the two protocols take relatively longer time to stabilize (the offered load in the 6th and 7th intervals remains the same as in 5th interval). On the other hand, ADV latencies do not change much after all the connections are established indicating that it stabilizes quicker.

ADV gives consistently lower latencies for the first five intervals because of the connection-initiation process carried out at the start of any new connection (see Section 3.1.1). This process is mainly intended to advertise the destination of the new connection as an active receiver; although as a side effect the routes to the destination are also known to all the nodes initially. With the destination node marked as an active receiver at all the nodes, future routing updates from the nodes would start refreshing the existing routes to the destination at all the nodes all the time.

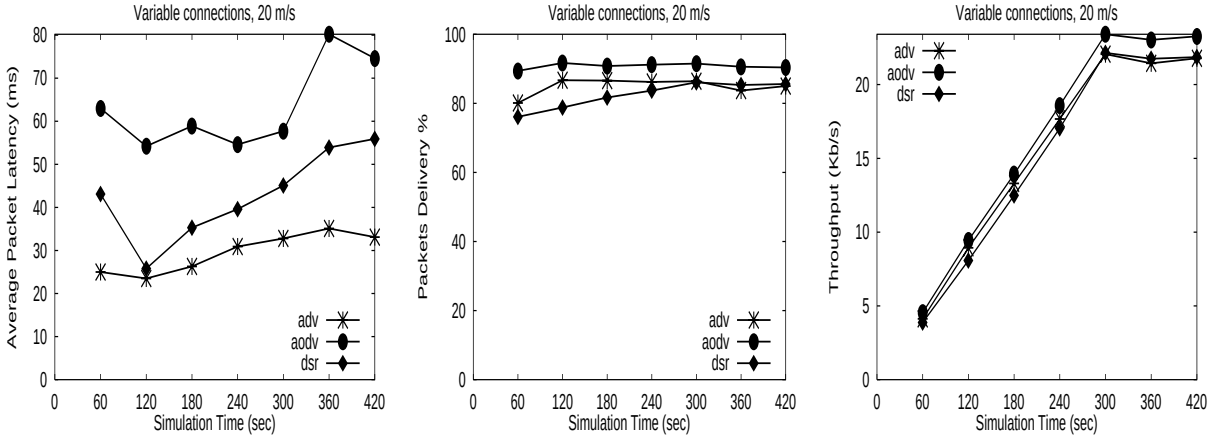

Figure 4.14: Average packet latencies, packet delivery rates and throughputs of ADV, AODV and DSR denoting the transient state conditions for the high speed 50 node square field.

Figure 4.15, gives the routing overhead in terms of packets/s and Kb/s. ADV transmits the least number of routing packets per second and also at a constant rate. AODV performs the worst of all the three with its overhead 3 times more than ADV particularly after all the connections have been established. As expected the routing load in terms of Kb/s is higher for ADV as the packet sizes of its updates are higher, carrying routes for all the active receivers which at the end of the simulation will be almost all the 50 nodes in the network. (Because of random source/receiver selection all the 50 nodes may not be active receivers.)

The MAC layer level overhead is as much as 50% lower for ADV than AODV and DSR for the same reasons given in the earlier sections; i.e., AODV and DSR transmit higher number of routing packets and more importantly has more number of unicast packets among the IP routing packets.

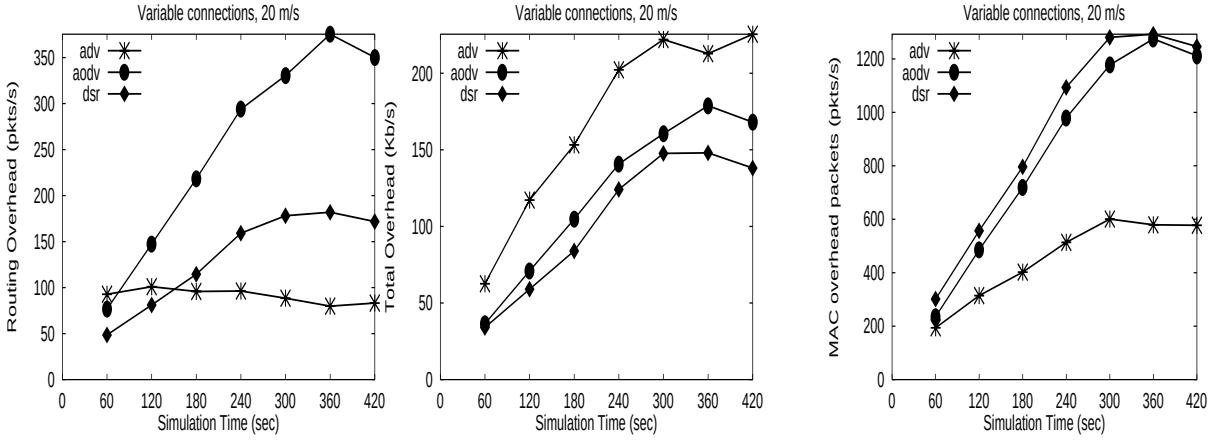

Figure 4.15: Routing overhead at IP and MAC layer level in terms of packets/s and in Kb/s of ADV, AODV and DSR denoting the transient state conditions for the high speed 50 node square field.

Low speed network

This section gives the performance differences for the three protocols in the low speed case. Figure 4.16 shows that ADV and DSR give very low latencies when compared to the AODV. The packet delivery rates are almost the same for all the three protocols except that ADV fails to match up for the last two intervals.

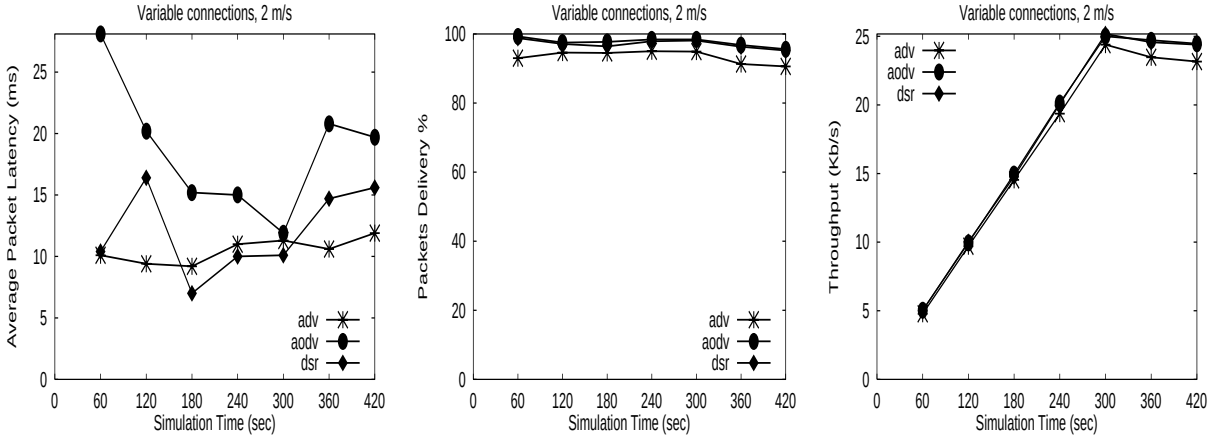

Figure 4.16: Average packet latencies, packet delivery rates and throughputs of ADV, AODV and DSR denoting the transient state conditions for the low speed 50 node square field.

From Figure 4.17, we can see that the ADV transmits high routing load in the transient state but stabilizes quickly for the last two intervals. This demonstrates the very good transient behavior of ADV.

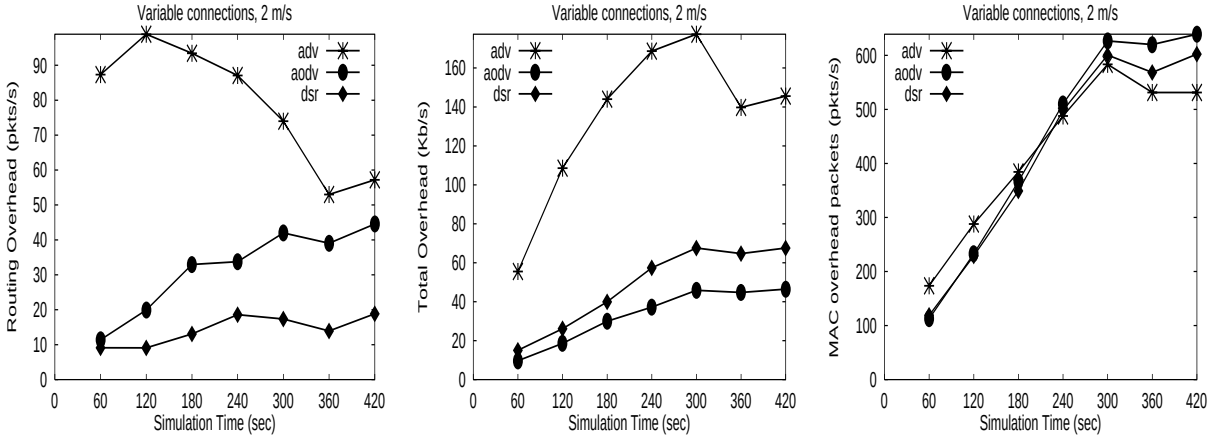

Figure 4.17: Routing overhead at IP and MAC layer level in terms of packets/s and in Kb/s of ADV, AODV and DSR denoting the transient state conditions for the low speed 50 node square field.

Observing the routing overhead in Kb/s, ADV transmits an increasing number of entries, as 10 new entries add to the network every interval, and stabilizes after all the new connections have been set up. Regarding the MAC layer level overhead, the same reason of high proportion of unicast packets among the routing packets of AODV and DSR explains the graph.

Static network

Figure 4.18, gives the packet latencies, packet delivery rates and throughputs in transient state conditions of all the three algorithms in the static network case. Once again, ADV and DSR give lower packet latencies compared to AODV. AODV gives high packet latencies initially, which can be attributed to the way route discoveries are done in AODV. In the expanding rings search the ring's radius is incremented by 2 each time a route reply is not received, which may take as many as 4 attempts to discover a receiver node. But in ADV, initially at the start of the connection, the source node propagates an init-connection packet advertising the destination as an active receiver.The receiver then sends an receiver-alert packet that sets up a route to the receiver at all the loads in the network immediately. Since in a static network the routes do not change, a quick initial connection establishment process for the receivers will help reduce packet latencies drastically. Also in DSR, the expanding rings technique first checks within one-hop radius. If a route is not found it broadcasts the route request network wide, thus taking only two

attempts for a route discovery. After the initial route discovery process the routing entries remain the same as the network does not change over time.

The packet delivery rates are 100% for all the protocols and the throughputs are also equal to the offered loads for all the three protocols.

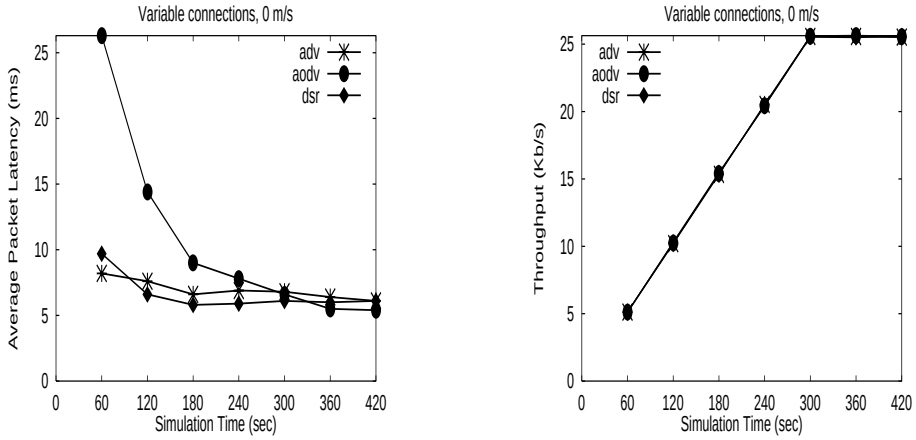

Figure 4.18: Average packet latencies, packet delivery rates and throughputs of ADV, AODV and DSR denoting the transient state conditions for a static 50 node square field.

Figure 4.20, gives the routing overhead in packets/s and Kb/s. ADV has the highest overhead in both the cases. As the network is static, a protocol should have a very low routing overhead at the time of connection establishment and virtually no overhead later when the routes do not change at all. But the 6th interval that should represent the zero overhead situation still transmits about 40 packets/s or 100 Kb/s. This implies that the static network is still seeing some neighbor changes. The reason lies in the way the “neighbor changes” variable is measured in ADV.

Referring to Figure 4.19, the nodes A and C are in the neighborhood of an active receiver B. Let the latest destination sequence number of B known to A is *x* and the metric is 1. Lets say, at some point of time, B transmits an update with a new sequence number *x+2*. Suppose that this update reaches only C and collides at A with another transmission from D (a hidden terminal). At a later time C transmits a routing update with B’s sequence number as *x+2* and a metric of 2, and suppose A promptly updates its B entry. Finally when B transmits another update (with a sequence number of *x+4*) and A receives

it directly and notices that the metric has changed from 2 to 1 and thus thinks that B has come into its 1-hop neighborhood. These type of scenarios still account for neighbor changes even though none of them take place and categorize the static network as a low speed network mostly. Thus it would trigger more updates and more such scenarios arise wherein neighbor changes are observed in a static network.

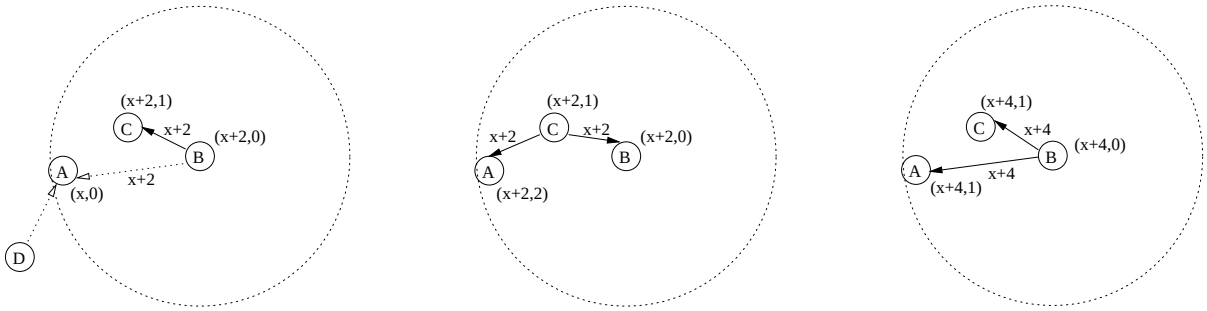

Figure 4.19: Scenario illustrations to explain the neighbor changes occurring even in a static network. (*m*,*n*) at a node defines that *m* is the latest destination sequence number of B known and *n* is the metric which gives the hop count to reach B. The solid arrows indicate that the update propagations are successful and the dotted arrows indicate that the update propagations are unsuccessful. The dotted circle indicates the transmission range of B.

It is to be noted that even though the number of routing packets transmitted are almost zero in AODV and DSR, there are still some routing bytes transmitted because of the various packet headers in the data packets. Also source routing overhead in DSR contributes significantly to the overall routing overhead. It is noteworthy that though AODV and DSR have very low overheads in packets/s after all the connections are set up. They still have measurable a 40 Kb/s overhead which is only 50% that of ADV.

The MAC overhead explains the almost same performance difference as the IP layer routing overhead. As the number of IP layer routing packets are less in AODV and DSR, the MAC layer level routing packets are soley because of the unicast propagations of data packets.

4.4 AODV and its optimization

As explained in section 3.2, we have introduced a new proactive routing technique into AODV, that we called BCAODV. In this section, we present the results of performance comparison of BCAODV with AODV for 20,40 and 60 connection cases in the high mobility 50-node square field at steady-state

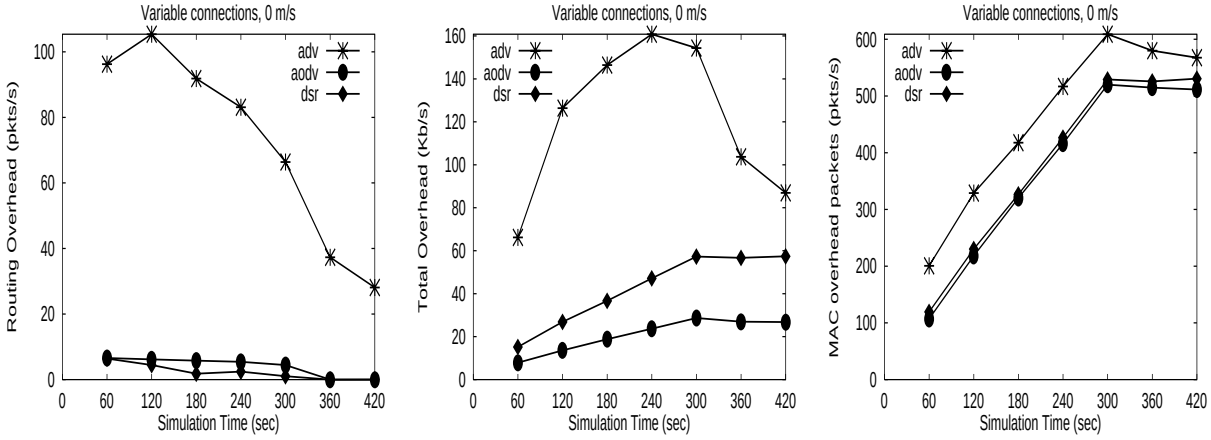

Figure 4.20: Routing overhead at IP and MAC layer level in terms of packets/s and in Kb/s of ADV, AODV and DSR denoting the transient state conditions for a static 50 node square field.

conditions. Later in the section the results for 40 connection low mobility 50-node square field are also presented.

4.4.1 Steady-state behavior at high speed

Average packet latencies. From Figure 4.21 we observe the fact that BCAODV improves the latencies by as much as 50% at low and moderate loads. This is because receiver-initiated beacons propagate fresher routes to all the nodes in the network periodically. This cuts down the need for additional route discovery process by the sources; thereby, cutting down on the packet waiting time. The reason for BCAODV not giving the same improvement at high loads, is that the routing overhead due to beacons is becoming relatively high, and thus making the wireless channel scarce for the data packets. Because of more number of retries to capture the channel, the data packets incur higher latencies.

Throughputs and delivery rates. Figures 4.22 and 4.23 give the throughputs and delivery rates of the two protocols respectively. We observe that, although AODV gives higher packet delivery rates (thereby higher throughputs) for loads greater than 60 Kb/s, the network is already in saturated conditions for those loads.

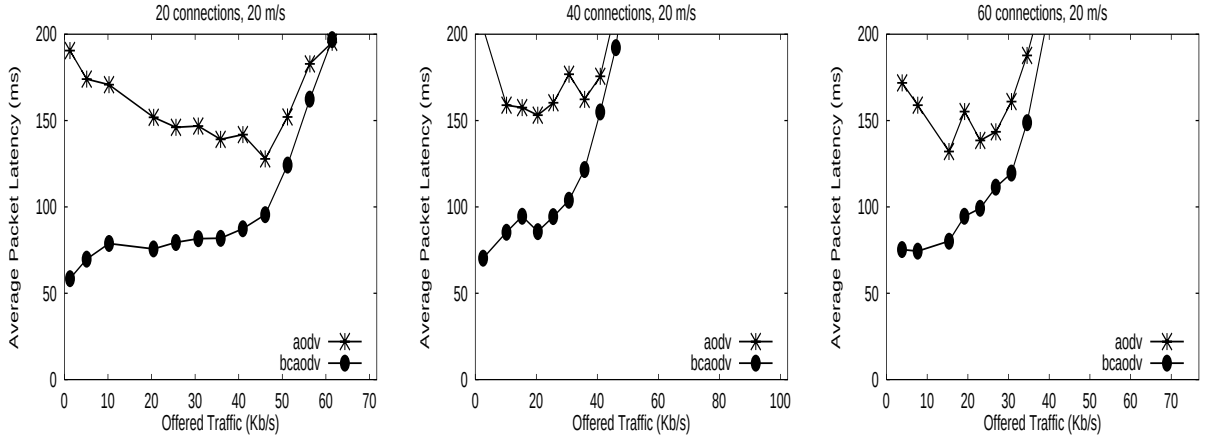

Figure 4.21: Data packet latencies of AODV and BCAODV for the high node mobility 50 node square field and various CBR connection cases.

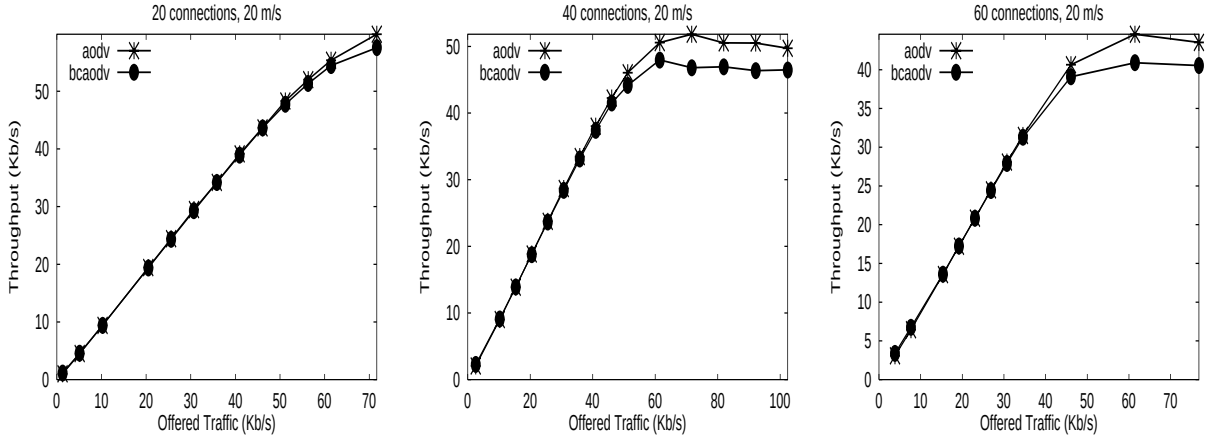

Figure 4.22: Data packet throughputs of AODV and BCAODV for the high node mobility 50 node square field and various CBR connection cases.

Routing packets per second. Figure 4.24 gives the routing packets transmitted per second for the two protocols. We do not see any significant additional routing packets because a node will atmost transmit one consolidated beacon packet per second. With the possibility that it can be piggybacked on route requests there will be little chance of separate consolidated beacon packets accounting for more routing packets. As the number of connections increase, AODV sends more routing packets than BCAODV. The more the number of connections, the more the number of route discovery processes. In BCAODV, as a single beacon can do the job of a route discovery process, we see less number of route discovery packets (less route requests and route replies).

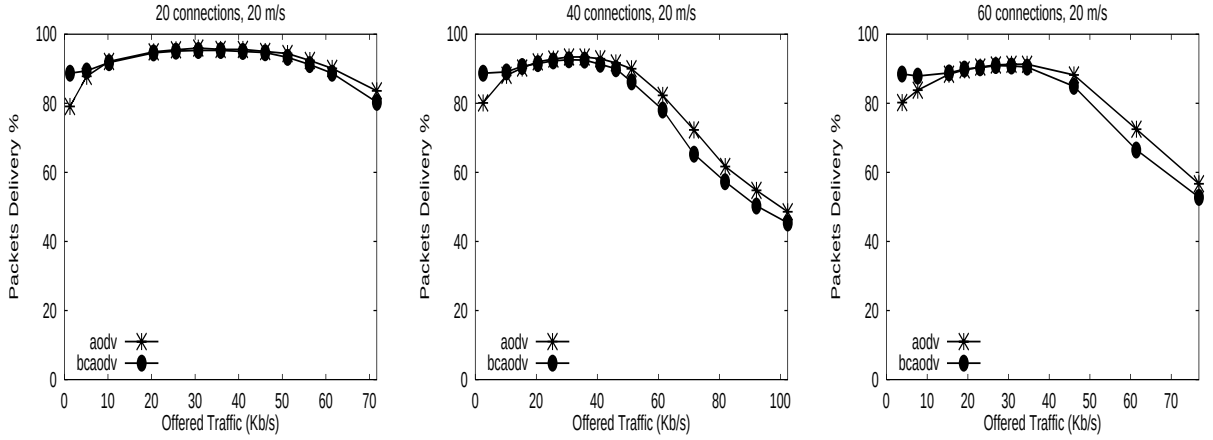

Figure 4.23: Data packet delivery rates of AODV and BCAODV for the high node mobility 50 node square field and various CBR connection cases.

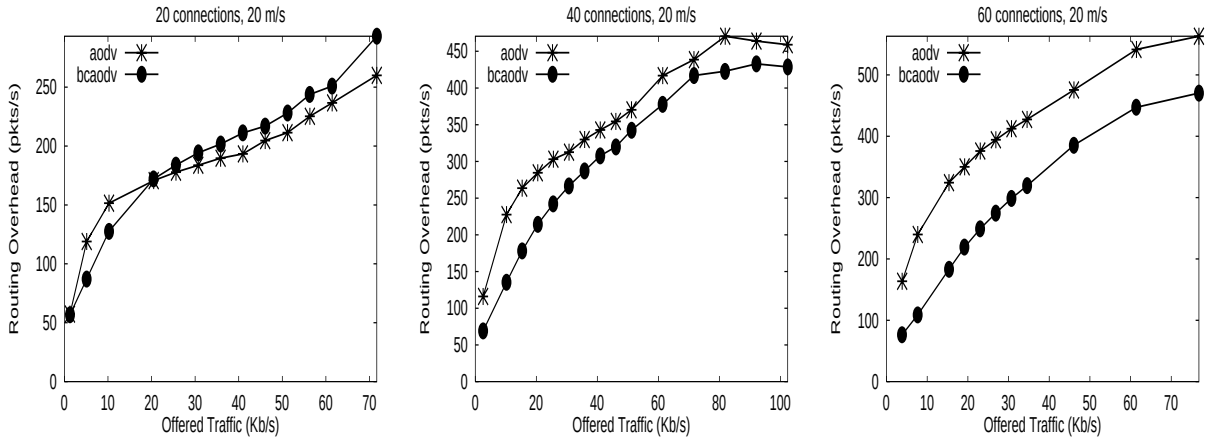

Figure 4.24: Routing overhead in packets/s of AODV and BCAODV for the high node mobility 50 node square field and various CBR connection cases.

Routing overhead in Kb/s. Obviously, the BCAODV transmits more routing bytes. This is because of the additional beacon entries propagated by either piggybacked on route requests or sent as separate beacon packets. Also, from Figure 4.25 we can see that, as the number of connections increase in the network the difference in routing overhead increases as BCAODV is forced to carry beacon entries of more receivers.

MAC layer level routing overhead. The routing packets transmitted at the MAC layer level for AODV and BCAODV are almost the same for the all connection cases. Any slight difference in the

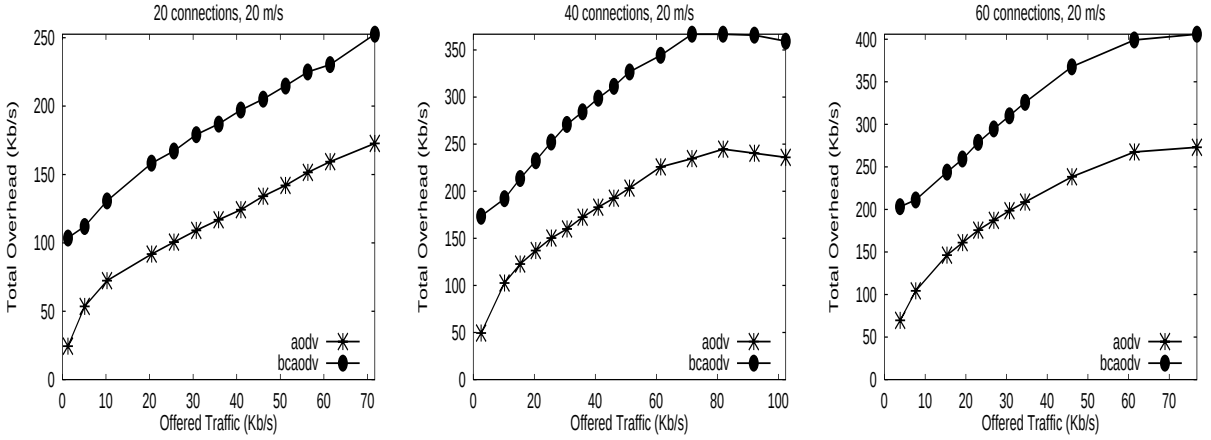

Figure 4.25: Routing overhead in Kb/s of AODV and BCAODV for the high node mobility 50 node square field and various CBR connection cases.

number of packets transmitted is mainly due to the increase in IP layer routing packets in the case of AODV for the 40 and 60 connections case.

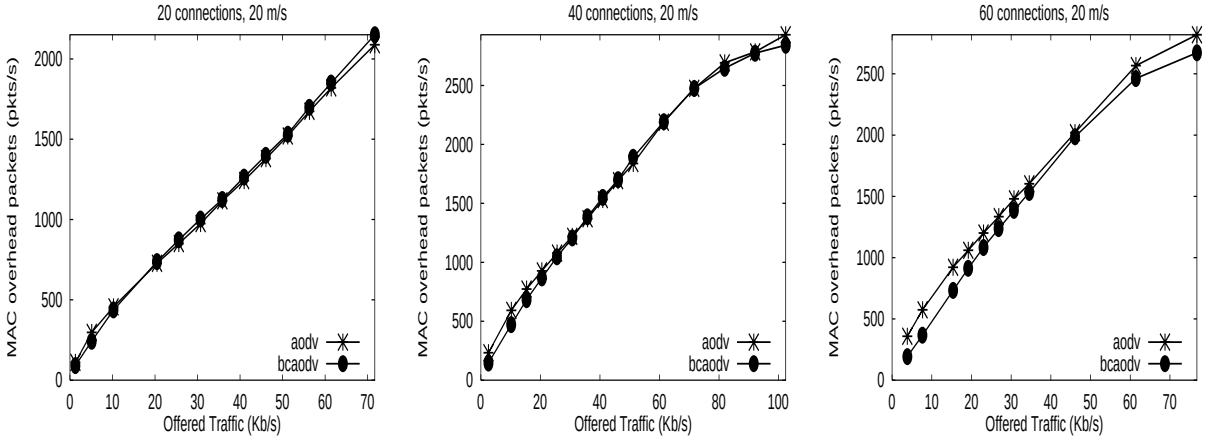

Figure 4.26: MAC layer level routing overhead in packets/s of AODV and BCAODV for the high node mobility 50 node square field and various CBR connection cases.

Additional Observations. An interesting observation that can be made is that the receiver-initiated beacons are not reducing the route discovery packets drastically. The route discovery packets include route requests and route replies. The reason may be that AODV is updating the routes too rapidly and that it is not able to use the beacons to its maximum extent. Also the way the beacons are propagated may take significant transmission time before they reach the source nodes, as the intermediate nodes

forward the consolidated beacon entries only at the end of their 1 second interval. Thus the immediate effective usage of the beacons may be minimized by the long propagation delay of the beacon entries.

4.4.2 Steady-state behavior at low speed

This section presents the performance comparison of various metrics between AODV and BCAODV protocols for the low speed mobile networks. The comparisons are done for 40 CBR connections for the low speed 50 node square field at steady-state conditions and shown in Figures 4.27 and 4.28. The results for 20 and 60 connections are almost similar and hence are not presented.

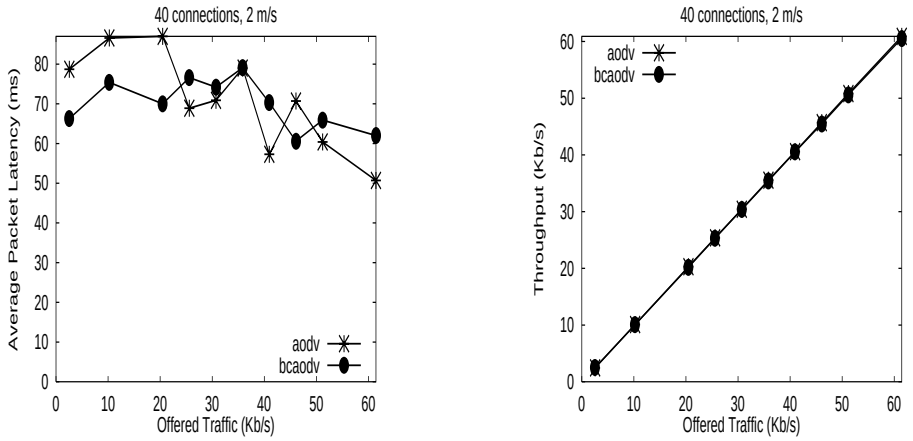

Figure 4.27: Data packet latencies and throughputs of AODV and BCAODV for the low node mobility 50 node square field and 40 connections case.

The average packet delays the throughputs offered by both the protocols are almost the same. This shows that the beacons propagated do not have much impact on reducing the latencies. As the number of route invalidations are less in low speed networks, the relative need for fresh routes decreases. But regardless of the need for routes, BCAODV maintains the same frequency with which the receivers transmit the beacons. Thus the beacon entries do not contribute much in reducing the packet latencies in low speed networks.

Observing Figure 4.28, we find that the IP overhead is much greater for the BCAODV protocol because of its beacon entry propagations. As the need for route discovery processes decrease in low

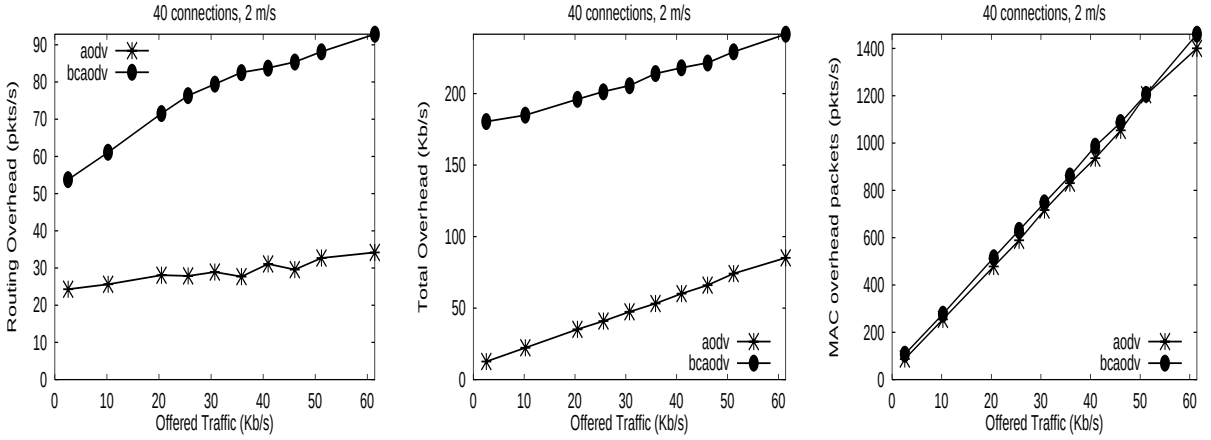

Figure 4.28: Routing overhead at the IP and MAC layers in packets/s and Kb/s of AODV and BCAODV for the low node mobility 50 node square field and 40 CBR connections case.

speed networks, the number of route discovery packets (route requests and route replies) reduce. As a result the consolidated beacon messages do not find sufficient route request packets to piggyback on and thus propagate as separate beacon packets. Thus the IP layer routing overhead is much higher in the case of BCAODV. The MAC layer level routing overhead is almost the same for both the protocols, with very slight increase in overhead in BCAODV accounting for the increase in the IP layer routing packets.

We can conclude from this section, that the beacon technique is not suited for low speed networks, as it will result in propagation of excessive overhead with no considerable reduction in delay. Reducing the frequency of the beacon interval may result in only a slight increase in overhead with a considerable reduction in latencies.

Chapter 5

Conclusions

Proactive routing protocols tend to provide lower latency than on-demand protocols because they try to maintain routes to all the nodes in the network all the time. But the flip side for such protocols is the excessive routing overhead transmitted. The routing updates are transmitted periodically with out much consideration for the network mobility or load. Depending upon the periodicity interval, the routing load can become excessive (insufficient) at low (high) loads.

To mitigate the effect of the periodic transmission of the updates, we have proposed some adaptive criteria that triggers routing updates based on network load and mobility conditions. The overhead is reduced by varying the size and the frequency of the routing updates dynamically. We call this new proactive protocol with on-demand characteristics, the Adaptive Distance Vector (ADV) protocol.

We have shown using simulations that ADV performs better than the on-demand protocols like AODV and DSR, especially in high speed networks. ADV provides the least latencies when compared to AODV and DSR at all the loads and can sustain loads far beyond the saturating points of the on-demand protocols. In terms of the routing overhead ADV transmits the least number of routing packets, although because of its larger size of routing updates it transmits more routing bytes compared to AODV and DSR (However, in a wireless medium, obtaining the channel for transmission is much more expensive in terms of power and network utilization than transmitting a few extra bytes with each packet). Moreover, considering the actual number of routing packets transmitted on the physical layer, i.e. MAC layer level routing packets, ADV transmits the least number of packets, thus utilizing the channel more efficiently.

AODV and DSR are not able to sustain high loads because of their large MAC layer level routing overhead that is actually transmitted on the physical medium. In AODV, the large number of route request packets account for most of the routing packets. In DSR, although the number of route request packets are less because of extensive use of snooping on source routes, the proportion of unicast routing packets (route replies and route errors) is a very high percentage (around 70% at average loads). This results in a large of number of MAC layer packets as unicast packets require RTS, CTS and ACK control packet exchanges.

The performance of ADV under low speed and infrequent node mobility networks is rather less encouraging. Although the routing overhead transmitted is substantial, the packet delivery ratios are less than expected. This is because the adaptive criteria is tuned to primarily handle frequent network changes. By fine-tuning the adaptive criteria, it is feasible to make ADV perform better at low mobility at the expense of somewhat higher routing overhead.

The good performance of the ADV in the transient conditions of a network at all speeds is noteworthy. The connection-initiation process at the start of any new connection, which advertises the destination as an active receiver, also enables quicker establishment of routes to the corresponding destination at all the nodes in the network. This helps ADV adapt to sudden load changes quickly and can be a good protocol in bursty traffic conditions.

We have also introduced a new proactive technique into AODV aimed at reducing the packet latencies. The technique involves broadcasting periodic receiver-initiated beacon messages which helps in propagating fresh routes to the receivers throughout the network. The simulations have shown that the latencies are reduced by as much as 50% in high speed networks. However, the beacons are not effective in low speed networks.

In summary, ADV is a strong candidate for the multi-hop mobile wireless environment. Since it combines both proactive and on-demand techniques, it exhibits the best characteristics of proactive algorithms and, at the same time, responsive to the network needs and conditions.

Future Work. In future, we would like to study the performance of ADV and the on-demand protocols for real-time traffic. With ADV providing lower latencies it should be a more suitable protocol for real-time traffic scenarios. It will be also interesting to investigate the effect of ADV and the on-demand protocols on TCP performance. As routes are frequently refreshed using updates in ADV, it helps maintain route connectivity all the time as required in TCP.

Appendix A

Performance Comparisons on 100 node rectangular field.

This Appendix presents the simulations done on 100 node rectangular field which is described in Section 4.1.2. The performance comparisons of ADV with on-demand algorithms in high speed and low speed networks are presented. The number of connections considered are same as before i.e 20, 40 and 60 connections with various packet rates. All the simulations are done until a protocol saturates which explains why some plots may look incomplete.

A.1 Steady-state behavior at high speed

Packet latencies. Referring to Figure A.1, ADV gives the least latency compared to the on-demand protocols at all loads. As the number of connections increase, the delay improvement is much more visible. Because ADV maintains routes to the destinations all the times, the data packets will have readily available routes to the destinations. The larger packet delays in the case of the on-demand protocols can be attributed to the larger route discovery time.

Packet delivery rates. As observed in Figure A.2, ADV gives consistently higher packet delivery rates for all the three cases. Though AODV performs well at low and average loads it exhibits a very drastic decline in the delivery rates after the offered load crosses around 60 Kb/s. DSR provides the least packet delivery rates for all the three cases at all the loads. The drastic degradation of performance

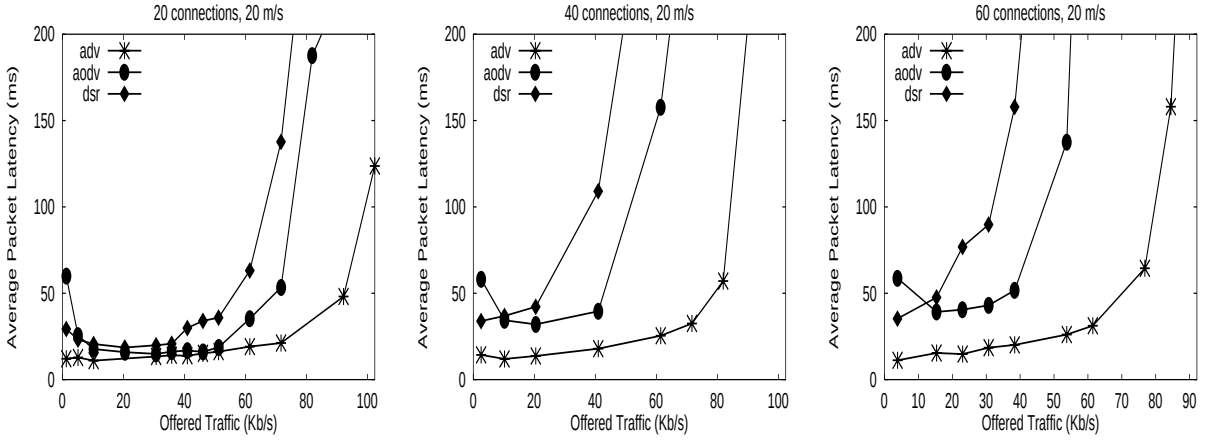

Figure A.1: Data packet latencies of ADV, AODV and DSR for the high node mobility 100 node rectangular field and various CBR connection cases.

of AODV is shown clearly in the 60 connections case when the network is loaded beyond the point of saturation. All the three protocols give almost the same packet delivery rates till their respective points of saturation.

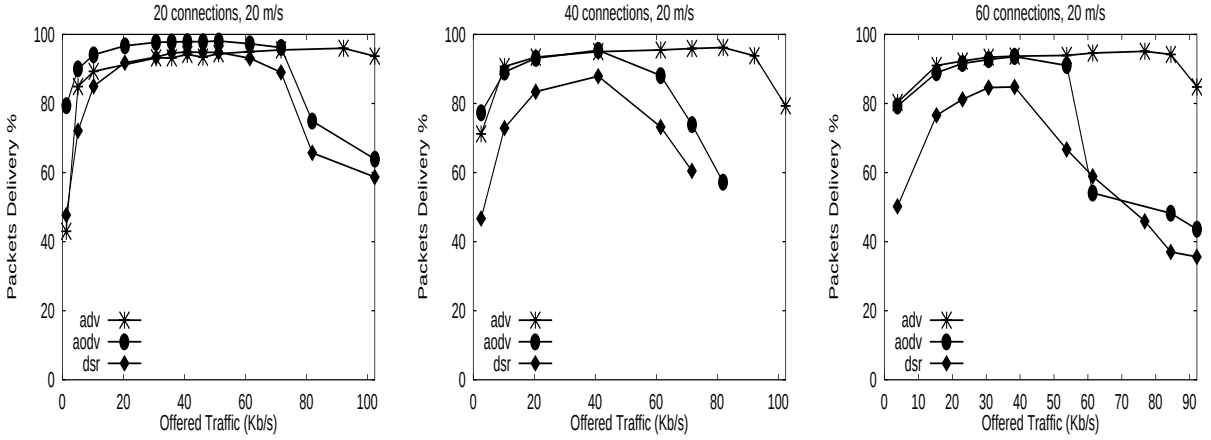

Figure A.2: Data packet delivery rates of ADV, AODV and DSR for the high node mobility 100 node rectangular field and various CBR connection cases.

Throughputs. Figure A.3 gives the throughputs offered by the three protocols. For 40 and 60 connections, ADV saturates at a load of 80 Kb/s; whereas, AODV and DSR saturate at only around 40 Kb/s. The reason for such poor performance of the on-demand protocols lies in the fact that the routing overhead reaches a critical limit; wherein, it starves the data packets for the physical medium. Also, as

the routing packets find fewer chances to propagate, the route discovery process takes longer time thus dropping some data packets (particularly at high data rates) from the fixed number of buffers at a node.

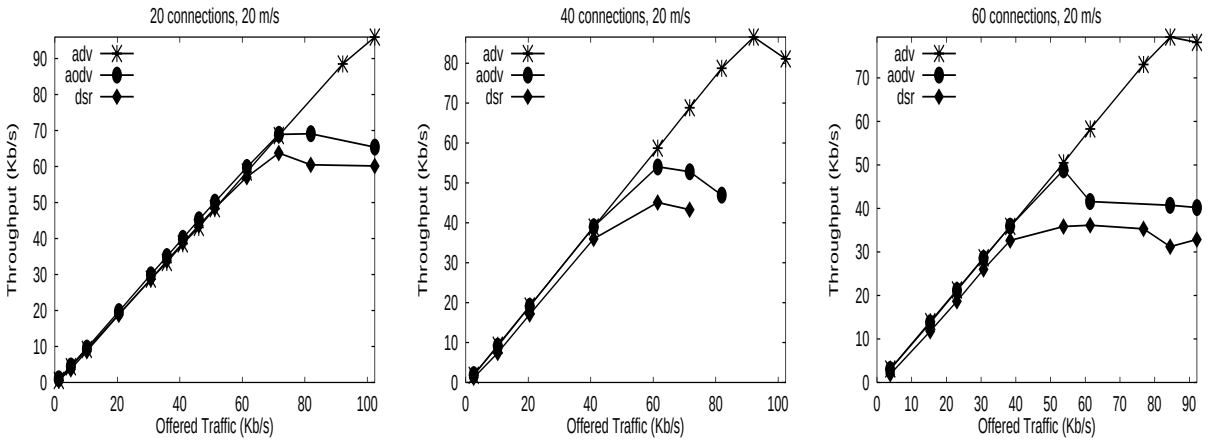

Figure A.3: Data packet throughputs of ADV, AODV and DSR for the high node mobility 100 node rectangular field and various CBR connection cases.

IP layer routing overhead. Figure A.4 gives the routing packets transmitted per second. In the 40 and 60 connections cases, ADV transmits the lowest number of routing packets followed by DSR and AODV respectively. However DSR transmits a very less number of routing packets than AODV, that can be explained by its use of route cache. Since this is a bigger field (2200mx600m) with more number of nodes than the one discussed in Chapter 4, the use of route cache is very helpful in reducing the number of route discovery processes, that often may take large boundaries. But as the number of connections increase DSR begins transmitting more number of route discovery packets that can be attributed to smaller cache size of 64 routes in total. (With multiple different routes stored for each destination, the cache size may still be inadequate for 40 and 60 connections.)

As expected ADV transmits the highest amount of routing bytes per second because of its larger routing packet sizes.

MAC layer routing overhead. This metric gives the actual number of routing packets transmitted on the physical medium. Figure A.6 shows how ADV, in spite of its higher packet delivery rates, gives the

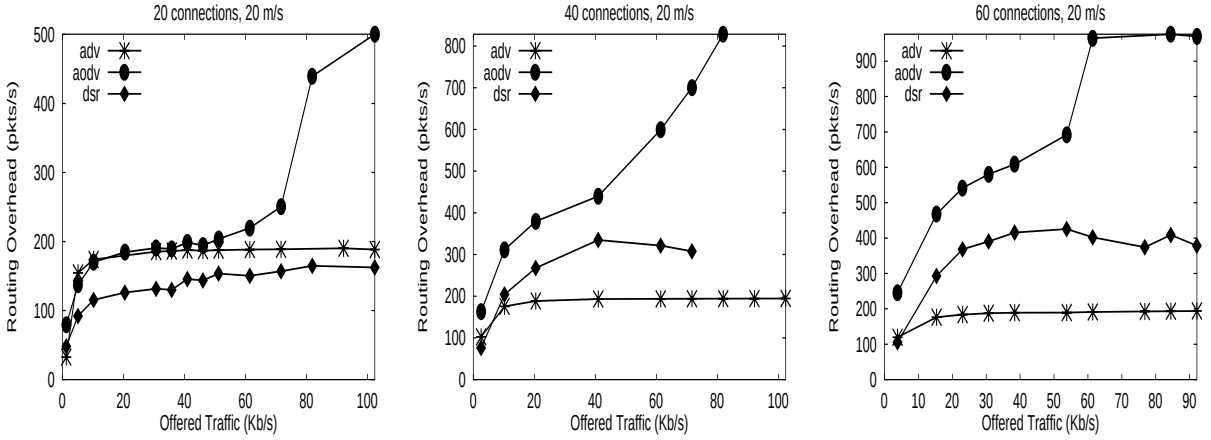

Figure A.4: Routing overhead in packets/s of ADV, AODV and DSR for the high node mobility 100 node rectangular field and various CBR connection cases.

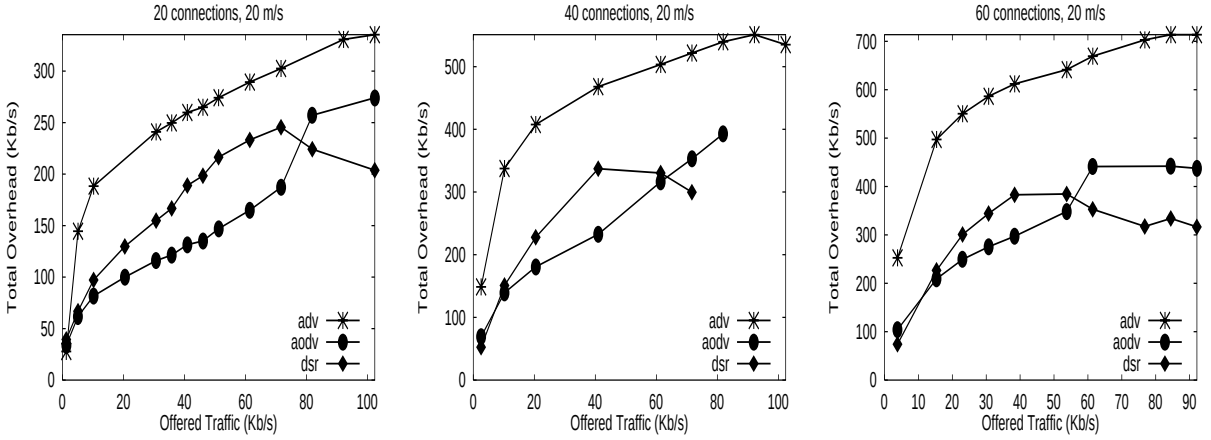

Figure A.5: Routing overhead in Kb/s of ADV, AODV and DSR for the high node mobility 100 node rectangular field and various CBR connection cases.

least MAC layer level routing overhead. AODV and DSR because of its higher IP layer overhead packets as well as the large proportion of the unicast routing packets contributes to the large MAC overhead.

A.2 Steady-state behavior at low speed

This section gives the performance comparisons of the three protocols for 40 connections and low speed network. The results for the 20 and 60 connections are almost similar and hence we chose to present only the 40 connections case. Figure A.7 gives the average packet latencies, throughputs and the delivery rates and Figure A.8 gives the routing overhead at the IP and MAC layer levels.

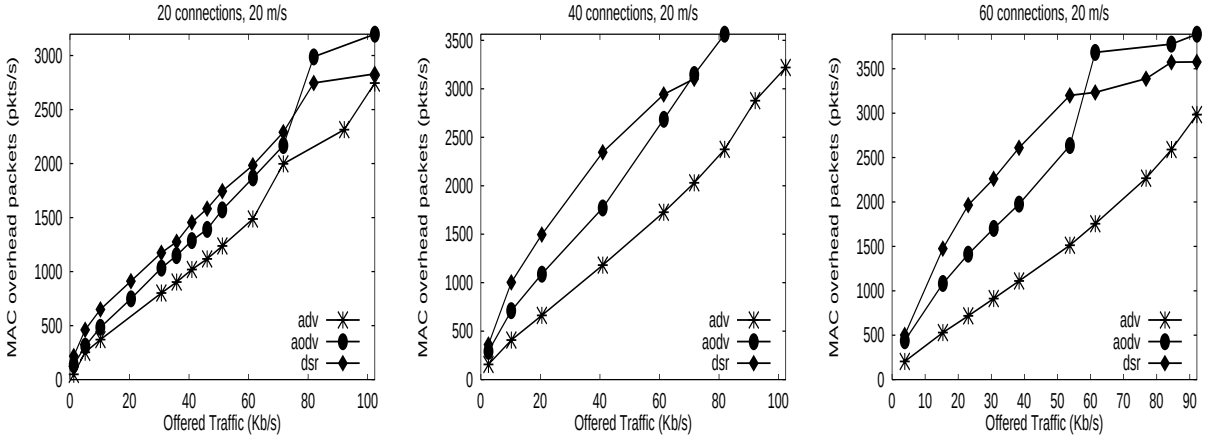

Figure A.6: MAC routing overhead in packets/s of ADV, AODV and DSR for the high node mobility 100 node rectangular field and various CBR connection cases.

The packet latencies, throughputs and the packet delivery rates offered by the three protocols are almost the same.

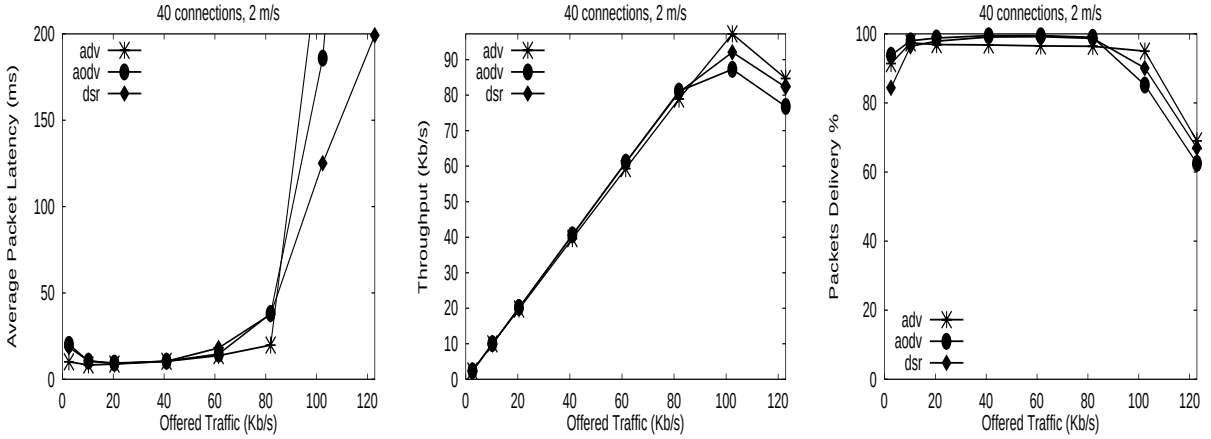

Figure A.7: Data packet latencies, throughputs and packet delivery rates of ADV, AODV and DSR for the low node mobility 100 node rectangular field and 40 CBR connections.

The IP layer routing overhead is the highest for the ADV protocol because of its proactiveness. The routing updates needed to maintain routes to all the active receivers all the time adds a significant amount of IP overhead. But seen at the MAC layer, the overhead transmitted by all the three protocols is the same. Thus ADV adds no extra overhead on the actual physical medium.

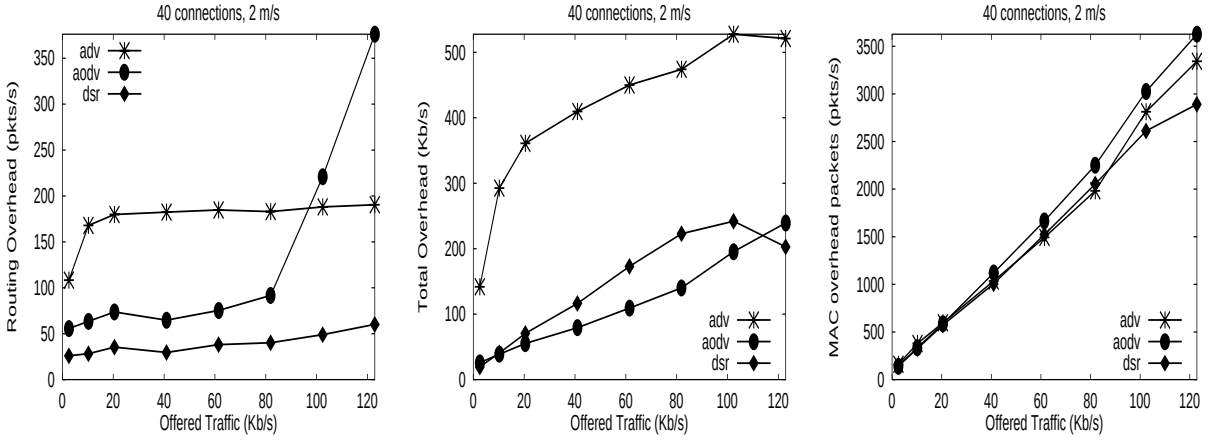

Figure A.8: Routing overhead at IP and MAC layer level in packets/s and Kb/s of ADV, AODV and DSR for the low node mobility 100 node rectangular field and 40 CBR connections.

Additional observations. The simulations for the 100 node rectangular field at high speed presented above almost represent the same behavior as with the 50 node square field presented in Chapter 4. As for the low speed network, ADV performs on par with the on-demand protocols unlike in the 50-node case where ADV fails to achieve higher packet delivery rates. This 100-node field better shows the drastic degradation of performance of AODV when the network is overloaded (beyond the point of saturation).

Bibliography

[1] J. Broch et al., "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *ACM Mobicom '98*, Oct. 1998.

[2] CMU Monarch Group, "CMU Monarch extensions to the NS-2 simulator." Available from http://monarch.cs.cmu.edu/cmu-ns.html, 1998.

[3] S. Corson and J. Macker, "Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations." RFC 2501, Jan. 1999.

[4] K. Fall and K. Varadhan, "NS notes and documentation." The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC. Available from http://www-mash.cs.berkeley.edu/ns, Nov. 1997.

[5] IEEE Computer Society LAN/MAN Standards Committee, "Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer (PHY) specification." IEEE Std. 802.3 4th ed., July 7,1993.

[6] IEEE Computer Society LAN/MAN Standards Committee, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications." IEEE Std. 802.11-1997. IEEE, New York, NY 1997.

[7] S. R. Das, C. E. Perkins, and E. M. Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *IEEE Infocom 2000*, Mar. 2000.

[8] D. B. Johnson et al., "The dynamic source routing protocol for mobile adhoc networks." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-02.txt, 1999.

[9] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad hoc on demand distance vector routing." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-03.txt, 1999.

[10] S. R. Das, R. Castaneda, and J. Yan, "Simulation based performance evaluation of mobile, ad hoc network routing protocols," in *Seventh Int'l Conf. on Computer Communication and Networks*, Oct. 1998.

[11] S.Lee, Mario Gerla, and C.K.Toh, "A simulation study of table-driven and on-demand routing protocols for mobile ad hoc networks," in *IEEE Network Magazine,* Aug. 1999.

[12] E.Royer, and C.K.Toh, "A review of current routing protocols for ad hoc mobile wireless networks," in *IEEE Personal Communications Magazine,* Apr. 1999, pp.46-55

[13] C. Cheng, R. Riley, and S. P. R. Kumar, "A loop-free extended Bellman-Ford routing protocol without bouncing effect," in *ACM SIGCOMM '89*, pp. 224–236, 1989.

[14] P. Johansson et al., “Scenario-based performance analysis of routing protocols for mobile ad-hoc networks,” in *ACM Mobicom ’99*, Aug. 1999.

[15] R.V.Boppana and S.R.Voona, “Designing efficient multicast protocols for mobile ad hoc networks”, Work in Progress.

[16] D. E. Comer, ed., *Internetworking with TCP/IP: volume 1*. Prentice Hall, Inc., 1995.

[17] J. J. Garcia-Luna-Aceves, “A unified approach to loop-free routing using distance vectors or link states,” in *ACM SIGCOMM ’89*, pp. 212–223, 1989.

[18] V. D. Park and S. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks,” in *IEEE INFOCOM ’97*, pp. 1405–1413, Apr. 1997.

[19] J. Moy, “Ospf version 2.” RFC 1247, July 1991.

[20] C. Hedrick, “Routing information protocol.” RFC 1058, 1988.

[21] P.Jacquet et al., “Optimized Link State Routing protocol (OLSR).” IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-01.txt, 2000.

[22] R.E.Bellman, *Dynamic Programming*. Princeton: Princeton Univ. Press, 1957.

[23] Z. J. Haas and M. R. Pearlman, “The zone routing protocol (ZRP) for ad hoc networks.” IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-zone-zrp-00.txt, 1997.

[24] M. Jiang, J. Li, and Y. C. Tay, “The cluster based routing protocol (CBRP) for ad hoc networks.” IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-cbrp-spec-01.txt, 1999.

[25] R. Sivakumar et al., ” CEDAR: Core Extraction Distributed Ad hoc Routing ,” in *IEEE Journal on Selected Areas in Communication, Special Issue on Ad hoc Networks, Vol 17, No. 8*, 1999.

[26] C. Toh, “Associativity based routing protocol (ABR) for ad hoc networks.” IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-longlived-adhoc-routing-00.txt, 1999.

[27] C. E. Perins and P. Bhagwat, “Highly dynamic destination-sequenced distance vector (DSDV) for mobile computers,” in *ACM SIGCOMM ’94*, pp. 234–244, Aug. 1994.

Vita

Satyadeva Prasad Konduru was born in Kakinada, India on April 13, 1977, the son of Jayasri Kumari and Raghava Prasad Konduru. After completing his schooling at Sri Venkateswara University School, Tirupati and N.St.Mathews Public School, Vijayawada he entered the Little Flower Junior College, Hyderabad, India, in 1992 for his college studies. He received the degree of Bachelor of Technology in Computer Science from Regional Engineering College, Warangal, India in May 1998. In August 1998, he entered the graduate program in Computer Science at The University of Texas at San Antonio.

Publications in Computer Science

Satyadeva P. Konduru and Rajendra V. Boppana,"On Reducing Packet Latencies in Ad Hoc Networks", to appear in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC) 2000*, September, 2000.

R. V. Boppana, M. K. Marina and S. P. Konduru, "An Analysis of Routing Techniques for Mobile and Ad Hoc Networks", in *Proceedings of the 6th International Conference on High Performance Computing (HiPC) 1999*, pp. 239-245, December, 1999.