DESIGNING EFFICIENT MULTICAST PROTOCOLS IN 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

DESIGNING EFFICIENT MULTICAST PROTOCOLS IN MOBILE AD HOC NETWORKS

by

SHANMUKHA RAO VOONA, 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

I would like to express sincere gratitude for the time, encouragement and guidance by Dr. Rajendra Boppana. I would also like to thank Dr. Robert Hiromoto and Dr. Richard Sincovec for serving on my committee and their suggestions. This thesis work has been partially supported by DOD / AFOSR grant F49620-96-1-0472.

August 2000

DESIGNING EFFICIENT MULTICAST PROTOCOLS IN MOBILE AD HOC NETWORKS

Shanmukha Rao Voona, B.Tech.
The University of Texas at San Antonio, 2000

Supervising Professor: Rajendra V. Boppana

Efficient multicast communication in MANETs (mobile and ad hoc networks) is important for interactive multimedia based communication among mobile computers. The hidden-terminal problem, in which packet transmissions to a common node from nodes that can not hear one another collide, can result in significant performance loss. Though this problem can be overcome using RTS / CTS (Request-To-Send / Clear-To-Send) handshaking mechanism for one-hop unicasts, there are no such solutions for reliable one-hop broadcasts which are crucial in designing efficient, high-performance multicast protocols.

To improve the reliability of one-hop broadcasts, we propose two forms of soft acknowledgement (soft-ack) schemes. We choose the soft-ack schemes instead of the TCP style hard acknowledgement schemes, to avoid excessive retransmissions. In our soft-ack schemes, we retransmit a packet at most 3-4 times, so that one-hop broadcast reliability is increased without excessive overhead or state information to be maintained in the nodes. Our soft-ack schemes can be applied in general to any MANET multicast or unicast communication protocol that requires highly reliable one-hop broadcasts. To demonstrate the benefits of our soft-ack schemes, we have enhanced ODMRP, a recently proposed multicast algorithm, with our soft-ack schemes.

We have simulated the original and the newer protocols under different network scenarios by varying traffic load, node speeds, number of data packet senders and different multicast group sizes. Our results indicate that the soft-ack schemes improve the delivery rate and the number of packet transmissions per delivered packet, without increasing packet latencies significantly.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A mobile and ad hoc network (MANET) facilitates mobile hosts, such as laptops with wireless radio devices, communicate among themselves even when there is no wired network infrastructure or central administration [5]. In a MANET, most hosts, if not all, are assumed to be moving continually and thus do not have a default router or fixed set of neighbors. Due to the limited radio propagation range of wireless devices, multiple hops may be needed to reach other mobile hosts. Therefore, each mobile host should have an Internet Protocol (IP) routing algorithm for building and maintaining routing tables, just like an internet router node. Such a protocol should support *unicasts*, *multicasts* and *broadcasts* efficiently. A unicast communication involves communication between a sender node and a receiver node while a multicast communication is from a node to a group of nodes in the network. A broadcast is a special case of a multicast and involves all the nodes in the network.

Currently, there are no major commercial applications that require such impromptu networking capabilities, but there is a real need for MANETs in military situations. Typical applications of MANET could include disaster recovery, crowd control, search and rescue, and automated battlefields.

Many of the these applications require close collaboration of teams (e.g., video / audio communication among team members). Since multicast communication may dominate unicast communication in such scenarios, multicasts should be supported efficiently. The hidden ter-

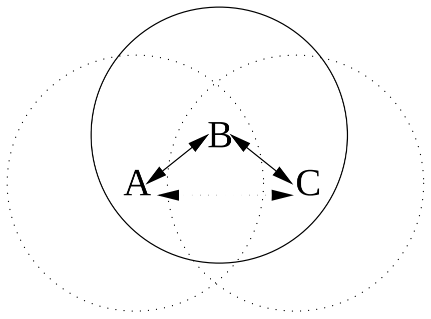

Figure 1.1: Hidden terminal problem. The circles indicate nodes' radio propagation ranges.

minal problem, described below, can cause significant performance degradation of multicast routing protocols for MANETs.

Problem description

The medium access control (MAC) protocol, with which mobile hosts can share a common broadcast channel, is essential in a MANET. The MAC protocol is generally a CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) protocol [12]. An example of such protocol is the IEEE 802.11 MAC [12] protocol. The CSMA/CA protocols attempt to prevent a station from transmitting simultaneously with other stations within its transmitting range by requiring each mobile host to listen to the channel before transmitting.

However, due to signal fading and attenuation, transmission of a mobile host may not be heard by another. As shown in Figure 1.1, B can communicate with both A and C, but A and C can not hear each other (or hidden from each other). Collisions can result if two stations hidden from each other, both believing the channel to be idle, try to simultaneously transmit to a common destination. This is the “hidden terminal” problem [30]. Unfortunately, the hidden terminal problem degrades the performance of a routing protocol substantially.

For reliable unicast communication between two neighbor nodes, the RTS-CTS handshake method is commonly used. When a sending station wants to transmit, it sends a request-to-send (RTS) to the receiver, who responds with a clear-to-send (CTS) if it receives the RTS correctly. RTS reserves the channel on the sender side and CTS reserves the channel on the receiver side. After a RTS/CTS exchange, the sender starts transmitting data reliably to the receiver. Between non-neighbor nodes, communication is achieved using multiple one-hop unicasts. So, providing efficient unicast communication is mostly a problem of knowing suitable routing paths to send packets, that has been extensively studied [23, 21, 14, 25, 10, 28].

Multicast communication among neighbors is achieved using one-hop broadcasts. Since the radio propagation range of a node is limited, multicast communication between non-neighbor nodes is achieved using multiple one-hop broadcasts, as in the case of non-neighbor unicast communication. A multicast routing protocol sets up these routes among multicast members (i.e. multicast sources and receivers) [24, 15, 3, 31, 18, 8]. However, the handshaking mechanism (RTS-CTS sequence) is not used for broadcasts and multicasts because multiple receivers (or neighbors) will have to respond with a CTS for a sender's RTS, that will cause severe channel congestion. Therefore, the hidden terminal issue is still a major problem for broadcasts and multicasts, and it significantly degrades the performance of a multicast routing protocol.

The impact of the hidden terminal problem on performance can be observed from Figure 1.2. To illustrate how a small % loss of packets, using one-hop broadcasts, results in dramatic reduction in overall delivery rate as the path length (between a multicast source and receiver) increases, we have computed delivery rates as a function of path length for different one-hop broadcast success rate and plotted them in Figure 1.2. A 3% loss in one-hop broadcast success rate, from 98% to 95% in the figure, results in 14% lower delivery rate after 5 hops. So, it is crucial to make the one-hop broadcasts highly reliable to achieve high delivery rates. To overcome the unreliable one-hop broadcasts problem, several researchers proposed mesh-based multicast schemes [15, 8] that use redundant paths to receivers to ensure high delivery rate. But this is not efficient, especially at high loads and in congested networks.

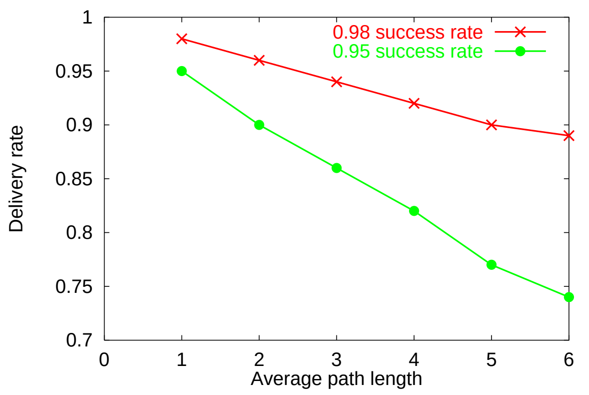

Figure 1.2: Delivery rate as a function of average path length, between a sender and a receiver, for different one-hop broadcast success rates. For a success rate of s and an average path length of p, the delivery rate is calculated as s^p.

Contributions of this thesis

To improve the reliability of one-hop broadcasts, we describe two forms of soft acknowledgement (soft-ack) schemes. In our soft-ack schemes, we retransmit a packet at most 3-4 times, so that one-hop broadcast reliability is increased without excessive overhead or state information to be maintained in the nodes. We have chosen soft-ack schemes rather than TCP-style hard acknowledgement schemes because TCP tries to achieve complete reliability by retransmitting unacknowledged packets many times before giving up. Enforcing such strict reliability can lead to excessive retransmissions that can result in congestion and collisions, resulting in degraded performance of protocols in MANETs. To demonstrate the benefits of these soft-ack schemes, we have applied them to ODMRP [15], a recently proposed multicast algorithm for MANETs. We have compared ODMRP with and without our techniques, using simulations of a network of 50 nodes moving randomly in a square field of 1000m x 1000m. We have tested the protocols under different network scenarios by varying traffic loads, node speeds, number of data packet senders and different multicast group sizes. Our simulations indicate that the delivery rate and the number of packet transmissions per delivered packet, can be improved significantly without increasing packet latencies significantly.

There have been other approaches to improve protocol performance by reducing the number of unnecessary transmissions in flooding (often used for control packets) [19, 13, 1, 20, 22]. Our acknowledgement schemes significantly differ from these approaches since we improve the reliability of one-hop broadcasts using soft-acks, that results in improved delivery rates.

Organization of the report

The rest of the thesis report is organized as follows: Chapter 2 gives a general overview of MANET and describes different multicast routing protocols proposed; Chapter 3 describes the techniques used for efficient multicast in MANETs; Chapter 4 describes the simulator and the underlying medium access (MAC) protocol used; Chapter 5 describes the simulation results; and finally, Chapter 6 concludes the whole report.

Chapter 2

Multicast Routing Protocols for MANETs

This chapter gives a brief overview of MANETs and describes different multicast routing protocols proposed so far.

2.1 Characteristics of MANETs

A MANET [5] consists of mobile platforms (e.g., a router with multiple hosts and wireless communications devices)–herein simply referred to as “nodes”–that are free to move about arbitrarily. MANET nodes are equipped with wireless transmitters and receivers using antennas that may be omni-directional (broadcast), highly-directional (point-to-point), possibly steerable, or some combination thereof. MANETs have several salient characteristics:

Dynamic topologies: Nodes are free to move arbitrarily; thus the network topology may change randomly and rapidly at unpredictable times, and may consist of both bidirectional and unidirectional links.

Bandwidth-constrained, variable capacity links: Wireless links will continue to have significantly lower capacity than their hardwired counterparts. In addition, the realized throughput of wireless communications–after accounting for the effects of multiple access, fading, noise, and interference conditions, etc.–is often much less than a radio's maximum transmission rate.

One effect of the relatively low to moderate link capacities is that congestion is typically the norm rather than the exception, i.e. aggregate application demand will likely approach or exceed network capacity frequently. As the mobile network is often simply an extension of the fixed network infrastructure, mobile ad hoc users will demand similar services. These demands will continue to increase as multimedia computing and collaborative networking applications rise.

Energy-constrained operation: Some or all of the nodes in a MANET may rely on batteries or other exhaustible means for their energy. For these nodes, the most important system design criteria for optimization may be energy conservation.

Limited physical security: Mobile wireless networks are generally more prone to physical security threats than are fixed-cable nets. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered. Existing link security techniques are often applied within wireless networks to reduce security threats. As a benefit, the decentralized nature of network control in MANETs provides additional robustness against the single points of failure of more centralized approaches.

A MANET routing protocol should be designed with the above characteristics in mind. Other desirable properties of a MANET routing protocol are: distributed operation; loop free routes; unidirectional link support; scalability in terms of the number of mobile nodes; and quality of service.

2.2 Multicast routing protocols

In this section, we describe flooding and the conventional multicast routing protocols designed for static networks, and state why they are not suitable for MANETs. We then describe the different mobile ad hoc multicast routing protocols proposed so far.

2.2.1 Flooding

Flooding can be used as a multicast protocol for MANETs. To prevent a chain reaction of data packet transmissions, we use a sequence number in each packet and increment it for each new packet. Each node keeps track of the set (Source IP, sequence number, group id) and processes only non-duplicate data. There are other approaches like Reverse Path Broadcast (RPB) that prevent a chain reaction of packet transmissions in flooding. In RPB, a node forwards the packet only if it received the packet from a parent node on the reverse shortest path to the source.

Flooding has been analyzed in [11] and compared with other multicast routing protocols [16, 20]. Flooding has a lot of redundant transmissions. In order to cut-down the number of redundant transmissions, many approaches have been suggested [19, 13, 1, 20, 22]. One of these approaches is to select a minimal set of one-hop neighbors to cover the set of all two-hop neighbors. In this way, the number of redundant transmissions is cut down.

Flooding has the advantage that it is very robust for mobility, since it uses a lot of redundant routes. But flooding has a large number of unnecessary transmissions that causes the protocol to quickly saturate, as compared with other multicast protocols.

2.2.2 Conventional routing protocols

The traditional Internet routing algorithms for multicast (e.g., CBT, PIM, DVMRP, etc.) are designed for static (mostly wired) networks. Core Based Trees (CBT) [2] is a protocol based on the concept of core nodes and a shared tree per multicast group. Any node, wishing to communicate with the multicast group, sends its packets to the core. The core, that is aware of the network topology, distributes the packets to the multicast group members. The use of core in mobile ad hoc networks is not efficient due to frequent node down times and continuous changing topology. Distance Vector Multicast Routing Protocol (DVMRP) [26] is a multicast protocol, that uses the distance vector distributed routing (DVR) algorithm in order to build a multicast tree for each multicast source. The Protocol Independent Multicasting (PIM) [7] is a hybrid protocol, that works in two modes related to CBT and DVMRP. These protocols are not suitable for MANETs since they are designed for a static topology and; therefore, have problems to converge to a steady state in an ad hoc network with a frequently changing topology.

2.2.3 Ad Hoc Multicast routing protocols

Recently many protocols have been proposed for multicast routing in MANETs [24, 15, 3, 31, 18, 8]. These protocols take into consideration the broadcast nature of the channel and continuous topology changes, and adapt themselves. Based on the routing structure used for multicast communication, these protocols can be classified as tree- or mesh-based protocols. A tree-based algorithm provides only a single path between any two multicast group members; whereas, a mesh-based algorithm provides one or more paths. Examples of mesh-based protocols are On Demand Multicast Routing Protocol (ODMRP) and Core Assisted Mesh Protocol (CAMP). Examples of tree-based protocols are Ad hoc On-demand Distance Vector (AODV) protocol, Ad hoc Multicast Routing (AMRoute) protocol, Ad hoc Multicast Routing protocol utilizing Increasing id-numbers (AMRIS) and Dynamic Source Multicast Routing (DSMR) protocol. In all of these protocols, one-hop broadcasts are extensively used for data packets and also for most control messages to maintain multicast tree or mesh. A performance comparison of four recent multicast protocols (ODMRP, CAMP, AMRIS, AMRoute) is presented by researchers at UCLA [16]. In the remainder of the section, we describe these algorithms.

On Demand Multicast Routing Protocol (ODMRP): ODMRP [15] creates a mesh of nodes, called the forwarding group, that forward multicast data packets via flooding within the mesh. ODMRP uses a soft-state approach in multicast group maintenance. Member nodes are refreshed as needed and they do not send explicit leave messages. ODMRP assumes bidirectional links between nodes.

Mesh Set up and Maintenance: In ODMRP [15], multicast sources set up and update group membership and multicast routes. A request phase and a reply phase comprise the protocol. A source that has multicast data to send but does not have routing or group membership information, floods a member-advertising-packet with data payload piggybacked. This packet, called “Join Query”, is periodically broadcasted to the entire network (every JQUERY_REFRESH seconds) to refresh the membership information and update the routes, as long as the source has multicast data to send. Though Join Query packets are sent periodically, ODMRP is considered to be an on-demand protocol because only active multicast sources send Join Query packets. The Join Query packet has the data payload piggybacked and contains the following routing information to establish routes: multicast group IP address; source IP address; unique sequence number; previous hop IP address; time to live (TTL); and hop count from source. When a node receives a Join Query packet, it stores the packet’s header, containing the routing information, in its “Message Cache”, for detecting duplicates. If the Join Query packet is not a duplicate, the node stores the packet in Message Cache and then puts its IP address in the previous hop field in the Join Query packet and increments the hop count. If the hop count is less than TTL, it broadcasts the Join Query packet. The previous hop address, stored in Message Cache, establishes the reverse path from the node to the source. This process sets up or updates the reverse paths to a source from each node. An example network is shown in Figure 2.1. Figure 2.1(a) shows how the network looks before multicast source S_1 sent Join Query packet. After S_1 sent the Join Query packet, the reverse paths from each node to S_1 are set up as shown in Figure 2.1(b).

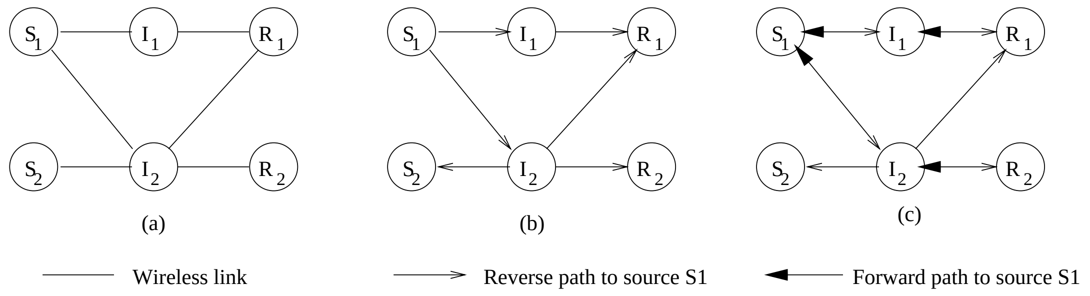

Figure 2.1: ODMRP Example (a) Before S_1 sent Join Query (b) After S_1 sent Join Query (c) After R_1,R_2 sent Join Reply

When a multicast receiver receives the Join Query packet, it creates and broadcasts a “Join Reply” packet to its neighbors. The Join Reply packet contains the multicast group address, a sequence of (source address, next hop address) pairs in that multicast group and a count of the number of pairs. The next hop IP address can be obtained from the previous hop address given in the Join Query packet header. When a node receives a Join Reply packet, it checks if the next hop address of one of the entries matches its own address. If it does, it sets the FG_FLAG (Forwarding Group Flag) for that multicast group address since it is on the path from a multicast receiver to a multicast source. It then builds a new Join Reply that is built upon matched entries. The next hop address for a source is obtained from the previous hop address in the Join Query message header from the source, stored in the Message Cache (Thus, the Message Cache also serves as a routing table). Then the node broadcasts the new Join Reply packet. These Join Reply packets have a TTL of 1 and are deleted by the node’s neighbors. This process propagates the Join Reply from a multicast receiver to a multicast source through the shortest delay path. Thus, this process sets up the route from a multicast source to each multicast receiver and creates a mesh of nodes, called the forwarding group. Multicast sources and receivers can also be a part of the forwarding group. In the example network shown in Figure 2.1, suppose that Join Query forwarded by I_1 reached receiver R_1 first and Join Query forwarded by I_2 reached receiver R_2 first. Figure 2.1(c) shows how the Join Reply packets from receiver R_1 and R_2 are propagated to multicast source S_1. Since I_1 is on the path from R_1 to

S_1 and I_2 is on the path from R_2 to S_1, both I_1 and I_2 set their FG_FLAG and thus become forwarding group members of the multicast group.

Forwarding group nodes, not refreshed within FG_FLAG_TIMEOUT seconds by Join Reply packets, are demoted to non-forwarding nodes. Source entries in Routing table entries, not refreshed within ROUTING_ENTRY_TIMEOUT seconds by Join Query packets, are removed. When a multicast source has data to send, it just broadcasts the data. When a node receives the data packet, it broadcasts the data packet if the data packet is not a duplicate and if the FG_Flag is set for that multicast group address.

ODMRP with Mobility Prediction: In networks where GPS (Global Positioning System) is available, ODMRP can be made adaptive to node movements by utilizing mobility prediction [15]. By using location and mobility information supplied by GPS, route expiration time can be estimated and receivers can select the path that will remain valid for the longest time. With the mobility prediction method, sources can reconstruct routes in anticipation of route breaks. In this way, the protocol becomes more resilient to mobility. The price is, of course, the additional cost and weight of GPS.

Advantages and Disadvantages: ODMRP can co-exist with any unicast routing protocol, since it finds its multicast routes independent of the unicast protocol used. ODMRP can also function as a unicast protocol. In ODMRP, unicast can be done using the multicast algorithm with a multicast group size of 2 and the destination IP address as the multicast group address. Multicast Data is broadcasted while unicast data is unicasted.

ODMRP has the typical advantage of mesh-based protocols. Mesh-based protocols are robust to mobility since they maintain and exploit multiple redundant paths. The redundant paths are helpful since packets can be delivered when normal links are broken due to mobility. These redundant paths are helpful at low and medium packet rates. At high packet rates, however, they turn out to be a disadvantage, since the additional packet transmissions on the redundant paths lead to congestion and collisions, and make the protocol to quickly saturate.

Another disadvantage of ODMRP is that the routing overhead increases as the number of senders increase.

Ad hoc on-demand distance vector (AODV): The AODV algorithm is designed to handle both unicast and multicast communication. The multicast operation of AODV is described as follows:

AODV [24] performs multicast by building and maintaining a multicast tree for each multicast group. Tree members can be multicast group members (MEMBER) or intermediate routers (ROUTER) connecting the group members. Each multicast group has a group leader (root of the tree). Each tree member has a parent (called an UPSTREAM neighbor) and zero or more children (called DOWNSTREAM neighbors). The group leader has no parent. Each node maintains a multicast routing table to keep track of multicast trees. There is one entry for each multicast group for which the node is a MEMBER or a ROUTER. A multicast entry contains next-hops (UPSTREAM and DOWNSTREAM neighbors), group address, group leader address, hop count to group leader, and group sequence number. A multicast group leader periodically broadcasts a Group Hello message to maintain the group connectivity.

When a node wishes to join the group, it broadcasts a Route Request (RREQ) packet with the join flag set. Only tree members respond to the request with a RREP packet. Other nodes rebroadcast the RREQ packet. Each node, on receiving the RREQ, adds the previous hop of that RREQ to the next-hops list for that group. If the sender of a RREQ receives multiple RREPs for its request, it chooses the best one based on the hop count and sequence number and sends a multicast activation (MACT) message. This message activates the path between the requesting node and reply node. If the source node does not receive a RREP within RREP-WAIT-TIME seconds, it retries up to RREQ-RETRIES times by re-broadcasting the RREQ with the broadcast id (used to distinguish multiple RREQs) increased by one. After RREQ-RETRIES, the node assumes that other tree members are unreachable and declares itself a leader.

Each node periodically broadcasts a non-propagating hello message, with TTL set to 1, to maintain local connectivity. When a tree link goes down, the DOWNSTREAM neighbor initiates a tree repair by broadcasting the RREQ with the repair flag set. It also sends the hop count information to the group leader. The tree repair process is similar to the join process with the exception that only tree members, with a hop count to the leader less than that of the source node, respond to the request. If the node that initiated the repair does not receive a RREP after several attempts (RREQ-RETRIES), it assumes the network is partitioned and follows the process, described below, that divides a tree into two sub trees.

If the node, that initiated the repair, is a ROUTER and has only one tree neighbor then it sends a MACT message with the prune flag set to that neighbor and prunes itself from the tree. If the ROUTER has more than one tree neighbor, it sends a MACT message with the group flag set to one of its neighbors that in turn propagates the request down the tree until a group member is reached. This node declares itself as the group leader of its portion of the tree and sends group hello message with the update flag set to indicate the change in the group leader info. All nodes receiving the group hello message with the update flag set, change their group leader information.

Two or more partitions of a multicast tree are joined into one by the join process [25]. The join process is initiated whenever the group leader with the lower IP address receives a group hello message from the other partition. The leader with the lower IP address sends a RREQ to the other group leader with both the join and repair flags set. The other leader then sends a RREP that activates the branch to this node. The two partitions are now combined into one. The leader with the higher IP address becomes the overall leader and sends the group hello message with the update flag set, indicating the change.

Advantages and Disadvantages: AODV can function as both unicast and multicast protocol. Since AODV is a tree-based protocol, it has the weakness of a typical tree-based protocol. In a tree-based protocol, tree links can break due to mobility and since there is no other route, packets are dropped until the tree is repaired. Since AODV has a shared tree routing structure, it has the advantage of constant routing overhead even when the number of senders increase (The multicast group size is constant in this case).

Core-Assisted Mesh Protocol(CAMP): CAMP [9, 8] supports multicasting by creating a shared mesh structure. All nodes in the network maintain a set of tables with membership and routing information. Moreover, all member nodes maintain a set of caches that contain previously seen data packet information and unacknowledged membership requests. CAMP classifies nodes in the network as duplex or simplex members, or non-members. Duplex members are full members of the multicast mesh, while simplex members are used to create one-way connections between sender-only nodes and the rest of the multicast mesh. “Cores” are used to limit the flow of JOIN REQUEST packets.

CAMP consists of mesh creation and maintenance procedures. A node wishing to join a multicast mesh first consults a table to determine whether it has neighbors that are already members of the mesh. If so, the node announces its membership via a CAMP UPDATE. Otherwise, the node either propagates a JOIN REQUEST towards one of the multicast group “cores”, or attempts to reach a member router by an expanding ring search of broadcast requests. Any duplex member of the mesh can respond with a JOIN ACK, that is propagated back to the source of the request.

Periodically, a receiver node reviews its packet cache in order to determine whether it is receiving data packets from those neighbors that are on the reverse shortest path to the source. If not, the node sends either a HEARTBEAT or a PUSH JOIN message towards the source along the reverse shortest path. This process ensures that the mesh contains all such reverse shortest paths from all receivers to all senders. The nodes also periodically choose and refresh their selected “anchors” to the multicast mesh by broadcasting updates. These anchors are neighbor nodes that are required to re-broadcast any non-duplicate data packets they receive. A node is allowed to discontinue anchoring neighbor nodes that are not refreshing their connections. It can then leave the multicast mesh if it is not interested in the multicast session and is not required as an anchor for any neighboring node.

CAMP relies on an underlying unicast routing protocol that guarantees correct distances to all destinations within finite time. Routing protocols that are based on the Bellman-Ford algorithm cannot be used with CAMP. Also, CAMP needs to be extended in order to work with on-demand routing protocols.

Advantages and Disadvantages: Since CAMP is a mesh-based protocol, it is robust to mobility by maintaining multiple redundant paths. Also, CAMP has good control traffic scalability for increasing multicast group size. Since JOIN REQUESTS only propagate until they reach a mesh member, CAMP does not incur exponential growth of multicast updates as the number of nodes and group members increase. However, it requires a unicast protocol to operate and is dependent upon the unicast protocol for behaviors regarding network convergence and control traffic growth in the presence of mobility.

Ad hoc Multicast Routing (AMRoute): AMRoute [3] creates a bidirectional shared multicast tree using unicast tunnels to provide connections between multicast group members. Each group has at least one logical core that is responsible for member and tree maintenance. Initially, each group member declares itself as a core for its own group of size one. Each core periodically floods JOIN-REQS (using an expanding ring search) to discover other disjoint mesh segments for the group. When a member node receives a JOIN-REQ from a core of the same group but from a different mesh segment, it replies with a JOIN-ACK and marks that node as a mesh neighbor. The node that receives a JOIN-ACK also marks the sender of the packet as its mesh neighbor. After the mesh creation, each core periodically transmits TREE-CREATE packets to mesh neighbors in order to build a shared tree. When a member node receives a non-duplicate TREE-CREATE from one of its mesh links, it forwards the packet to all other mesh links. If a duplicate TREE-CREATE is received, a TREE-CREATE-NAK is send back along the incoming link. The node receiving a TREE-CREATE-NAK marks the link as mesh link instead of tree link. The nodes wishing to leave the group send the JOIN-NAK to the neighbors and do not forward any data packets for the group.

Advantages and Disadvantages: The key characteristic of AMRoute is its usage of virtual mesh links to establish the multicast tree. Therefore, as long as routes between tree members exist via mesh links, the tree need not be readjusted when network topology changes. Non-members do not forward data packets and need not support any multicast protocol. Thus, only the member nodes that form the tree incurs processing and storage overhead. AMRoute relies on an underlying unicast protocol to maintain connectivity among member nodes and any unicast protocol can be used. The major disadvantage of the protocol is that it suffers from temporary loops and creates non-optimal trees when mobility is present.

Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS): AMRIS [31] establishes a shared tree for multicast data forwarding. Each node in the network is assigned a multicast session ID number. The ranking order of ID numbers is used to direct the flow of multicast data. Like ODMRP, AMRIS does not require a separate unicast routing protocol.

Initially, a special node called Sid broadcasts a NEW-SESSION packet. The NEW-SESSION includes the Sid's msm-id (multicast session member id). Neighbor nodes, upon receiving the packet, calculate their own msm-ids that are larger than the one specified in the packet. The msm-ids thus increase as they radiate from the Sid. The nodes rebroadcast the NEW-SESSION message with the msm-id replaced by their own msm-ids. Each node is required to broadcast beacons to its neighbors. The beacon message contains the node id, msm-id, membership status, registered parent and child's ids and their msm-ids, and partition id. A node can join a multicast session by sending a JOIN-REQ. This JOIN-REQ is unicasted to a potential parent node with a smaller msm-id than the nodes's msm-id. The node receiving the JOIN-REQ sends back a JOIN-ACK if it is already a member of the multicast session. Otherwise, it sends a JOIN-REQ.PASSIVE to its potential parent. If a node fails to receive a JOIN-ACK or receives a JOIN-NAK after sending a JOIN-REQ, it performs "Branch Reconstruction (BR)." The BR process is executed in an expanding ring search until the node succeeds in joining the multicast session.

AMRIS detects link disconnection by a beaconing mechanism. If no beacons are heard for a predefined interval of time, the node considers the neighbor to have moved out of radio range. If the former neighbor is a parent, the node must rejoin the tree by sending a JOIN-REQ to a new potential parent. If the node fails to join the session or no qualified neighbors exist, it performs the BR process.

Data forwarding is done by the nodes in the tree. Only the packets from the registered parent or registered child are forwarded. Hence, if the tree link breaks, the packets are lost until the tree is reconfigured.

Advantages and Disadvantages: Since AMRIS has a shared tree routing structure, AMRIS is sensitive to mobility. Other negative aspects in AMRIS are the number of transmissions and the size of beacons. Beacons can cause a number of packet collisions even when nodes are stationary. In more dense networks, the performance may become even worse. Also, the selection of Sid can affect the shape of the tree and possibly its performance.

Chapter 3

Techniques for Efficient Multicast in MANETs

The hidden terminal problem, described in Chapter 1, results in a significant performance degradation of multicast routing protocols in MANETs. To overcome the hidden terminal problem, we propose acknowledgement-based techniques to do efficient multicast; and thereby, improve the performance of the multicast protocol. These techniques can be applied in general to any mobile ad hoc communication protocol that requires highly reliable one-hop broadcasts. As an example, we have applied our techniques to the On-Demand Multicast Routing Protocol (ODMRP).

ODMRP is a mesh-based protocol. Mesh-based protocols have a lot of unnecessary data transmissions overhead. Therefore, we first describe a technique to cut down the unnecessary data transmissions. Then, we describe the acknowledgement-based techniques used for efficient multicast.

3.1 Reducing data transmissions

ODMRP creates a mesh of nodes, called the Forwarding Group (FG), that forward multicast data packets via flooding within the mesh. An example mesh created by ODMRP is shown in Figure 3.1. S_1, S_2 are the multicast sources. R_1, R_2 are the multicast receivers. F_1, F_2, F_3, F_4

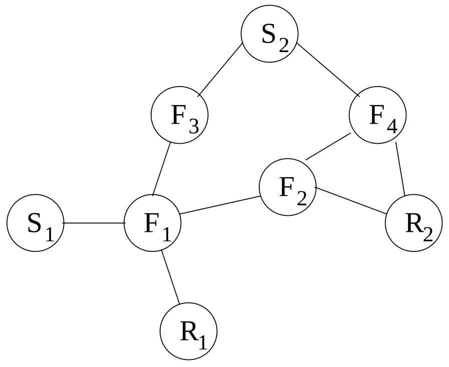

Figure 3.1: An example mesh created by ODMRP. S_i are sources, R_i are receivers, F_i are forwarding nodes. The nodes within the radio range of a node are indicated by lines.

are the FG members. F_1, F_2 are included in the FG due to source S_1 and F_1, F_3, F_4 are included in the FG due to source S_2.

When a node receives a data packet, it forwards the same if (a) it is a FG member for the multicast group and (b) has not received this data packet before; otherwise, the node discards the packet. In the example, Figure 3.1, if data transmissions (broadcasts) are reliable, the following data forwarding pattern occurs when S_1 broadcasts a packet: F_1 receives the packet and forwards (rebroadcasts) it; S_1, R_1, F_2, F_3 receive the forwarded packet from F_1; F_2 and F_3 forward the packet since they see it for the first time and their FG flags are set; S_1 and R_1 do not rebroadcast since they are not in the FG set; S_2, F_1 receive the packet from F_3 and R_2, F_1, F_4 receive the packet from F_2; F_1 ignores the duplicate packets, while S_2 and R_2 do not rebroadcast since they are not in the FG set; F_4 forwards the packet; S_2, R_2, F_2 receive the packet from F_4; and finally, F_1 ignores the duplicate packet while S_2 and R_2 do not rebroadcast as they are not in the FG set.

If data transmissions (broadcasts) are reliable, only the FG members, created due to source (say S_1), need to forward S_1's data. If a FG member of other source receives that data packet and forwards it, it is an unnecessary data transmission. The redundant routes, created by these other sources' FG members, are helpful when normal routes are broken due to mobility or when a collision due to hidden terminal problem occurs. In the above example, the forwarding of S_1's data packet by F_3 and F_4 is redundant and results in unnecessary data transmissions overhead, if F_2's broadcast is received by R_2. However, if R_2 did not receive the packet from the expected node F_2 (due to either collision or F_2 moving away from R_2) and received the packet from F_4 then F_4's transmission would be helpful in delivering the packet to R_2. This method may increase the throughput by transmitting data packets on every possible route in the mesh, at the expense of a large number of redundant data transmissions. At low packet rates, the impact of this large number of redundant data transmissions is not seen. But at high packet rates, this results in contention and makes the protocol quickly saturate, resulting in low delivery rate and high delay.

To cut down these unnecessary data transmissions overhead, we maintain the FG flags on a per sender basis. Only FG members of a source, say S_1, forward S_1's data packets. This cuts down the number of data packet transmissions. We call this new protocol *OMP-ps* (On-demand Multicast Protocol-per-sender). Compared to the original ODMRP, the proposed modification improves packet latencies and possibly reduces the throughput, since redundant transmissions are cut down.

To improve any loss in throughput, we have made the one-hop broadcasts more reliable by using soft acknowledgement (soft-ack) schemes. These techniques differ from the TCP-style hard acknowledgement schemes that try to achieve complete reliability by retransmitting unacknowledged packets many times before giving up. Enforcing strict reliability, as in TCP, can lead to many retransmissions, resulting in congestion and collisions and; thereby, decreasing the network throughput and increasing packet latencies. Instead, we use soft-ack schemes that may retransmit a packet at most 3-4 times. Such soft-ack schemes improve the reliability of one-hop broadcasts without causing excessive overhead or state information to be maintained in the nodes. The improvements in the reliability of one-hop broadcasts result in improved delivery rates and throughput.

2-bit Status Code	Interpretations
00	the node received the packet, or it does not need to report the status.
01	the node is reporting NAK for the first time
11	the node is repeating NAK; no one responded to previous NAK(s).
10	the node received the packet after sending one or more NAKs earlier.

Table 3.1: Status Codes used in the negative-acknowledgement technique

3.2 Negative-Acknowledgements (NAKs)

In the NAK technique, when a FG member or receiver of a multicast group receives a data packet, it reports to its neighbors whether it has the previous data packets (from the source where the received packet originated) or not. Based on this information, neighbors respond by retransmitting any missing data packets. This lets multicast group members use redundant paths in the network (not just the multicast mesh) when needed to recover data packets lost either due to collision or mobility. Below, we describe the NAK technique in detail.

Before sending a data packet into the network, a multicast source stamps it with a data sequence number. Successive data packets from a multicast source have sequence number in increasing order (sequence numbers are used by all multicast algorithms). To incorporate the Negative-Acknowledgement technique, we add a 1-byte status code to each data packet. This status byte reports on the status of the preceding four data packets (having sequence numbers $n-1$, $n-2$, $n-3$, $n-4$ if this data packet has sequence number n) using 2 bits per packet. The 2-bit status codes and their interpretations are given in Table 3.1. The importance of the different values for the status is explained later.

When a source generates data packets, it clears the status byte to 0. Before forwarding a data packet, a node updates the status byte to report its own status of the previous four data packets. FG members keep track of the first data sequence number (say m) they received since they became a FG member. And status bits in status byte are set to 00 for data packets having sequence number lesser than the stored data sequence number m. This is done so that FG members do not report NAK for data packets sent when they are not in the FG mesh. Since some receivers may not need to forward data packets (e.g., R_1 and R_2 in Figure 3.1), they can indicate packets' status by broadcasting a packet containing only the 1-byte status code (i.e. a data packet with zero data payload). Since this contributes to additional packet transmissions and also makes the region around the receivers contention-prone and error-prone due to collisions with normal packets, receivers send the status code packets only when they have to report a NAK (i.e. if any 2-bit status in the status code has 01 or 11 as its value). In order to respond to NAKs, we require *all* nodes, not just the mesh nodes, to store the 5 most recent data packets received from each (multicast source, multicast group).

When a node sends a NAK, all of its neighbors may try to respond to the NAK. This can lead to multiple responses (or multiple retransmissions of the NAK'ed packet). To mitigate this, we use the following optimizations:

- Only FG members of the source, that sent the data packet, or the source can respond to status code of 01 (i.e. NAKs reported for the first time). For status bits of 11, *any neighbor* node (not necessarily a mesh node) that has a copy of the NAK'ed packet can respond. As mentioned earlier, to facilitate the NAK scheme, all nodes keep track the 5 most recent packets received from each source of each multicast group.

- Child nodes do not propagate NAKs of their parent nodes (i.e. if parent node puts 01 or 11 in the status bits, then the child nodes change the status bits to 00 while reporting their status flag, even if they do not have the packet). This is done mainly to reduce the number of NAK responses. To see this, let us consider a simple example. Suppose node x did not receive a data packet and it reports a NAK in the next data packet. All the nodes on the path from x to the receivers will not receive the data packet, with very high probability. If all of these nodes report a NAK and the neighbors of these nodes send responses to these NAKs, it can lead to many NAK responses. To reduce excessive NAKs, only node x issues a NAK. When a neighbor node responds to that NAK, it will forward the packet. This solution is not completely reliable as the NAK response data packet may get lost on its way from node x. Our objective is not to provide complete reliability, instead we want to provide efficient multicasts, i.e., multicasts with high throughput and low latency. If we insist on complete reliability, the network will become contention-prone and will have reduced throughput and higher average latency. We do not insist on 100% reliability, since some loss is acceptable with video/audio transmissions. In any event, the upper layer (TCP and the application layer) protocols should deal with providing complete reliability, if necessary.

Since we use broadcast for transmitting all data packets, all neighbors of a node can hear its transmissions. Figure 3.2 outlines the different steps followed by a node, when the node receives a data packet (or a status code packet).

We apply this NAK technique only for data packets. Join Query packets, that carry both data and routing information, use flooding. For this reason, we do not use NAK technique for Join Query packets as it causes more overhead. Join Replies are transmitted using unicast. We applied this NAK technique to *OMP-ps* and call this new protocol *OMP-n* (On-demand Multicast Protocol-NAK).

3.3 Passive-Acknowledgements (PACKs)

The NAK technique significantly improves the throughputs of ODMRP and OMP-ps protocols. But it also increases the average packet delay because all the NAK response data packets have an additional delay of at least one inter-packet arrival interval. The packet delay is increased because a lost packet is NAK'ed when the next data packet is sent. To reduce the delay, we add the PACK technique to the OMP-n protocol. Below, we describe the PACK technique

```
Procedure Receive-data-packet(Packet $P$)
// When a node receives a data packet or a status code packet, this procedure is invoked.

{
    Check-NAK-retransmission-cache-and-IFQ($P$)
    // IFQ (or interface queue) is the queue that holds
    // packets awaiting transmission by the network interface.
    Respond-to-NAK($P$)
    if ( this node is in the FG of the source, where $P$ originated, or is a receiver
        && $P$ is a normal data packet (i.e. not a status code packet)
        && $P$ is not a duplicate (i.e. $P$ is received for the first time))
    {
        Update the status code in $P$ to report this node's status of the previous four data packets.
        Store $P$ in the recently received data packet cache.
        Check-status-code-of-IFQ-packets($P$)
        if ( this node is a receiver && this node is not in the FG of the source, where $P$ originated)
            Remove the data payload from $P$. // status-code packets do not carry data payload.
        if ( $P$ is a NAK response packet )
            Place $P$ in front of IFQ, so that it is forwarded immediately.
        else
        {
            Forward $P$ with a jitter of 10 msec.
            Normally all packets are sent after a short random delay (jitter)
            between 0 and 10 msec, in order to prevent synchronized collisions.
        }
        // If $P$ is a NAK response, we forward $P$ immediately so that neighbors
        // can receive this packet quickly and refrain from sending duplicate NAK responses.
    }
}
```

```
Procedure Respond-to-NAK(Packet $P$)
// In this procedure, a node, that received $P$, responds to the NAKs present in $P$.

{
    for all NAKs (i.e. if 2-bit status, in the 1-byte status code, is 01 or 11) in $P$
    if (((2-bit status is 01 && this node is in the FG of the source, where $P$ originated) ||
        (2-bit status is 11)) && this node has the desired NAK'ed packet)
    {
        Check if there is already a copy of the packet in the NAK retransmission cache
            or the interface queue(IFQ) and if not present, place a copy of the packet in
            NAK retransmission cache.
            // NAK retransmission cache contains packets to be retransmitted
            // in response to NAKs and these packets are sent to the network with a
            // inter-packet time-gap of random(5) msec.
    }
}
```

Figure 3.2 Continued on next page

Figure 3.2 Continued from previous page

```
Procedure Check-NAK-retransmission-cache-and-IFQ(Packet $P$)
// In this procedure, a node, that received $P$, uses heuristics to remove NAK responses from
// NAK retransmission cache and interface queue(IFQ). These removals may not be totally
// correct and are done in order to reduce the number of NAK responses

{
    Check if copy of $P$ is present in NAK retransmission cache and remove it if present.
    Check if copy of $P$ is present in IFQ and remove the copy if it is a NAK response.
    if ( the 1-byte status code of $P$ has any 2-bit status as 10 )
    {
        Check if a copy of that previous data packet, for which the 2-bit status is meant,
            is present in NAK retransmission cache and remove that copy.
        Also check in IFQ and remove the copy if it is a NAK response.
    }
}
```

```
Procedure Check-status-code-of-IFQ-packets(Packet $P$)
// Packets may not come in order and also due to NAK response packets, packets in IFQ
// may not be indicating the correct status code. This procedure checks the IFQ packets
// and corrects the status code if the status code indicates the status of $P$.

{
    Check every data packet (including status code packets) in the IFQ and if its status code
        indicates the status of $P$, correct the status if necessary.
    Also after correction, if any status code packet no longer indicates a NAK (i.e. if the
        1-byte status code does not contain any 2-bit status as 01 or 11), remove that packet.
}
```

Figure 3.2: Pseudocode of the different steps followed at a node, when the node receives a data packet (or a status code packet). In the pseudocode, we sometime use C and C++ style coding. Procedure names in procedure calls are italicized. ”If” and ”for” are in bold to imply conditional statements.

in detail.

A FG node has other FG nodes or receivers as its neighbors. Receivers can also be FG nodes. When a node, say x, forwards a data packet, it expects the neighbor FG node, say y, to forward that packet. Since all data transmissions are broadcasts, node x can hear node y forwarding its packet and can treat this as a passive acknowledgement. Because of hidden terminal problem, x may not be able to hear the retransmissions from all neighbor FG nodes. If we require node x to hear all neighbor FG nodes to forward its packet, node x may have to retransmit many times to fulfill the condition. Instead of rigidly requiring that x should hear all neighbor FG nodes' retransmissions, we only require node x to hear at least one neighbor FG node forwarding the packet. If a neighbor of node x is a receiver-only node, it will not retransmit the data packet. As in the case of NAKs, we could require receivers to send an explicit ACK for x to hear it. However, this is not desirable since there will be too many ACKs (that also make the area near a receiver contention- and collision-prone) and also, the benefit is not that significant. Instead, we require that if any receiver is present in a node's neighborhood, the node should not expect a passive acknowledgement.

When a node, say x, sends a data packet and there is no receiver in its neighborhood, it stores the packet in a PACK retransmission cache. When x hears a passive acknowledgement, it removes the packet from the PACK retransmission cache. If after sending a packet, the node does not receive a passive acknowledgement within 30 msec, then the node retransmits once and removes the packet from the retransmission cache. Retransmitting more than once may not be fruitful. This is because if the neighbor FG node has actually forwarded the packet after the first transmission and node x did not hear the forwarding due to hidden terminal problem then even if node x retransmits, the neighbor node will not forward (since it considers the retransmitted packet as a duplicate). Also, retransmitting more than once will increase the number of retransmissions and decrease the performance. We have tried with different maximum retransmission counts and found that retransmitting once results in a better performance than others.

At low packet rates, the PACK technique combined with the NAK technique is likely to reduce packet delay compared to the NAK-only technique. This is due to the reduction of the number of NAK response data transmissions. For high packet rates, PACK technique increases the number of collisions and results in nodes not able to hear their neighbors forwarding. This results in a lot of retransmissions and increases the delay instead of decreasing it. Therefore, we have implemented a congestion-based PACK technique. This technique is described below.

Each node monitors the channel activity. If a node senses the channel busy, or if it has one or more packets present in MAC layer IFQ, it backs off from sending any PACK retransmissions. What we mean by backing off is that we change the waiting time for sending the retransmission in PACK retransmission cache to the CURRENT_TIME + 30 msec. By backing off due to packets in IFQ, we are giving preference to normal packets and NAK responses. These packets are essential and PACK retransmissions are not that essential because NAK technique will take care of any missing packets, when the next packet is sent. A PACK retransmission is cancelled if the next data packet from the source is received.

So in order to send a PACK retransmission, the channel should be idle for continuous 30 msec from the time the normal packet was sent. When packet rate is low, PACK retransmissions will be sent. But when the packet rate is high, very few PACK retransmissions will be sent because the channel is busy for most of the time. This way, at low packet rates, the average packet latencies decrease compared to the NAK-only scheme, and at high packet rates, the same average delay is maintained as that of NAK-only technique.

We applied this PACK technique to *OMP-n* and call the new protocol as *OMP-np* (On-demand Multicast routing Protocol-NAK-PACK).

Our NAK and PACK techniques are similar to the Negative and Positive Acknowledgement schemes described for the Internet [27, 17]. The NAK technique proposed for the Internet [27], generates separate NAK packets. Since additional packets increase the overhead and can cause performance degradation, we piggyback NAKs on data packets. Our NAK scheme differs from the proposed NAK scheme for the Internet in the following aspects: child nodes do not propagate parent's NAKs; only FG members of the source that sent the data packet, or the source can respond to first time NAKs; and we use some heuristics to remove duplicate responses to the NAK'ed packet.

The Positive Acknowledgement technique proposed for the Internet [17], requires all receivers to send separate acknowledgements to the sender and creates more overhead compared to passive ACKs. The Positive Acknowledgement technique for the Internet and a similar PACK scheme suggested in the ODMRP draft [15], require acknowledgements from all receivers. In order to reduce the number of retransmissions and since the neighborhood of a sender frequently changes, our PACK technique requires only one of the neighbors' retransmission. Other points of difference are: nodes retransmit unacknowledged packets only once; a node does not expect an acknowledgement if any receiver is present in its neighborhood; and our PACK technique is a congestion-based technique.

Chapter 4

Simulation Environment

We have used the Network Simulator 2 (*ns-2*) [6] from UC Berkeley for our simulation studies. To simulate the mobile wireless radio environment, we have used the mobility extension [4] to *ns-2*, developed by the CMU Monarch project at Carnegie Melon University. The version of CMU extensions, we used, is 1.1 (released on August 12, 1998). This version uses the 2.1b1 version of *ns-2* (released on November 11, 1997).

4.1 Network Simulator

Network Simulator 2 is the result of an on-going research and development at the University of California, Berkeley. *ns-2* 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 OTcl (Object Tool Command Language). *ns-2* uses an OTcl interpreter towards the user. This means that the user writes an OTcl script that defines the network (number of nodes, links), the traffic in the network(multicast sources, receivers, type of traffic) and which protocols it will use. This script is then used by *ns-2* 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).

The version of *ns-2*, we used, does not support mobile wireless environment. So, we used the mobility extensions to *ns-2* developed at CMU.

4.2 Mobility extension

The CMU mobility extension adds the following features to the Network Simulator:

Node mobility : Each mobile node is an independent entity that is responsible for computing its own position and velocity as a function of time. Nodes move around according to a movement pattern specified at the beginning of the simulation.

Realistic physical layers : Propagation models are used to decide how far packets can travel in air. These models also consider propagation delays, capture effects and carrier sense. We use the Two Ray Ground Reflection Approximation (Uses Friss free-space attenuation $1/r^2$ at near distances and an approximation to Two Ray Ground $1/r4$ at far distances).

MAC 802.11 : An implementation of the IEEE 802.11 Distributed Coordination Function (DCF) MAC [12] protocol was included in the extension. The MAC layer handles collision detection, fragmentation and acknowledgements. 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.

The “RTS / CTS / Data / ACK” pattern is used for unicast packets for reliable communication. 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 times until this ACK is received. DCF is designed to use both *physical carrier sense* (i.e. sensing the channel to determine if it is busy) and *virtual carrier sense* mechanisms to reduce the probability of collisions due to hidden terminals. By setting timers based upon the reservations in RTS/CTS packets, the virtual carrier sense augments the physical carrier sense in determining when mobile nodes perceive that the medium is busy. Broadcast packets are sent only when virtual and physical carrier sense indicate that the medium is clear, but they are not preceded by an RTS/CTS and are not acknowledged by their recipients. Therefore, for broadcast packets, there is the possibility of collisions due to hidden terminal problem.

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).

Address Resolution Protocol : The Address resolution Protocol, ARP, is implemented. ARP translates IP-address to hardware MAC addresses. this takes place before the packets are sent down to the MAC layer.

Ad-hockey : Ad-hockey is an application that makes it possible to visualize the mobile nodes as they move around and send/receive packets. Ad-hockey can also be used as a scenario generator tool to create the input files necessary for the simulations. This is done, by positioning nodes in a specified area. Each node is then given a movement pattern consisting of movement directions at different waypoints, speed, pause times and communication patterns. Example snapshots of nodes' positions, generated by ad-hockey for node mobility speeds of 10 m/s, at different simulation times (0, 100, 200 secs) are shown in Figure 4.1.

Radio network interfaces : This is the model of the hardware that actually transmits the packet onto the channel with a certain power and modulation scheme.

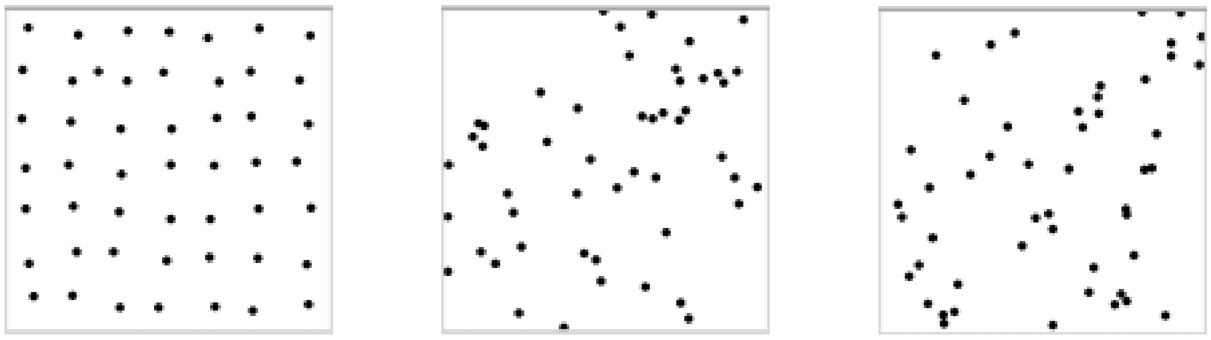

Figure 4.1: Snapshots of nodes' positions, generated by ad-hockey at 0, 100, 200 simulation seconds. The dots represent the nodes in the 1000m x 1000m simulation field. Number of nodes is 50 and node speed is 10 m/s.

Transmission power : The radius of the transmitter is about 250 meters in this extension. An omni-directional antenna is used.

4.2.1 Shared media

The extension is based on a shared media model (Ethernet in the air). This means that all mobile nodes have one or more network interfaces that are connected to a channel as in Figure 4.2. 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 and reception of packets on another channel. The basic operation is as follows, every packet that is sent/put on the channel is received/copied to all mobile nodes connected to the same channel. When a mobile nodes receive a packet, it first determines if it is possible for it 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. Detailed description of what happens when a packet is sent or received, is given in the next subsection.

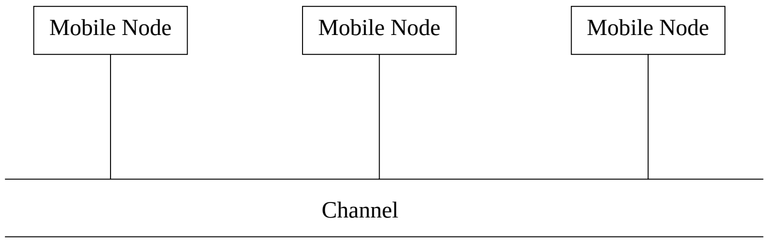

Figure 4.2: Shared media model.

4.2.2 Mobile Node

Each mobile node, as shown in Figure 4.3, makes use of a routing agent (protocol) for the purpose of calculating routes to other nodes in the ad hoc network. Packets are sent from the application and are received by the routing agent. The agent decides a path that the packet must 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 decides it is ok to send it onto the channel, it fetches the packet from the queue and hands it over to the network interface that 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. Each network interface stamps the packet with the receiving interfaces properties and the invokes the propagation model.

The propagation model uses the transmit and receive stamps to determine the power with which the interface will receive the packet. The receiving interface then use its properties to determine if it successfully received the packet, and sends it to the MAC layer if appropriate. If the MAC layer receives the packet error- and collision- free, it passes the packet to the mobiles 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,

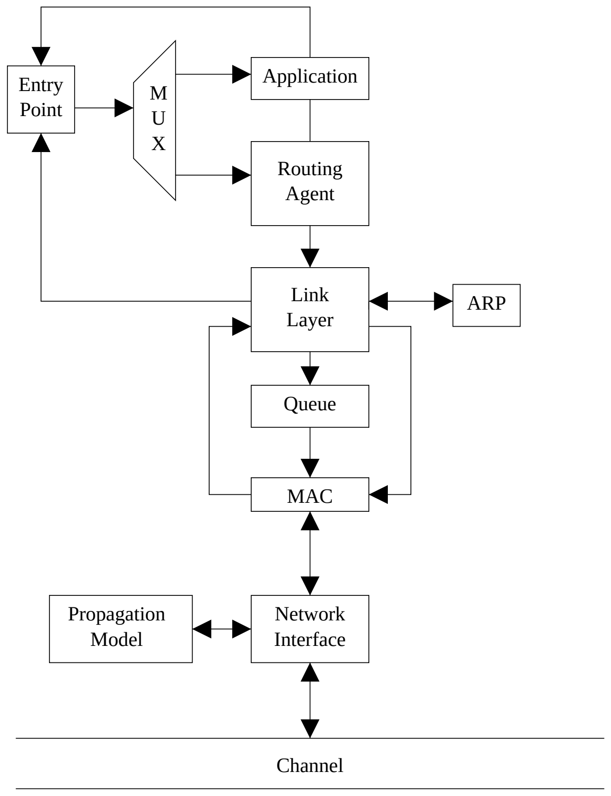

Figure 4.3: A mobile node.

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.

4.3 Implementation Details

We have implemented ODMRP in *ns-2* for our simulation studies. The implementation is according the new ODMRP draft [15]. A few major issues are noteworthy.

- We have not assumed GPS capability for our mobile nodes.
- The timers used for all ODMRP simulations are given in Table 4.1
- Join Reply propagation: A Join Reply packet contains the multicast group id and a list of

Parameter	**Value**
JQUERY_REFRESH Interval	3 sec
FG_FLAG_TIMEOUT	10 sec
ROUTING_ENTRY_TIMEOUT	10 sec

Table 4.1: Various timers used in the ODMRP implementation.

(source address, next hop address). Whenever a receiver receives a Join Query packet, it creates a Join Reply containing entries for all the multicast sources of that multicast group. When the number of sources increase, this method of Join Reply packet creation results in a large number of Join Reply packets sent in the network. A Join Query packet updates the reverse path routes to the source that generated it. So, it is only necessary that the receiver send the Join Reply packet to the source that sent the Join Query packet. This way, the Join Reply updates the multicast routes only to the source that sent the Join Query packet and eliminates unnecessary Join Reply propagation to all other multicast sources. We have tested both the methods and found that the second method has a better performance than the original method. Also, the authors of ODMRP draft suggested the second method when the number of sources is large [29]. In our simulations, we want to test how the protocol performs for a large number of senders. So in our implementation, we follow the second method and send the Join Reply only to the source that sent the Join Query. Since this Join Reply packet contains only a single entry of (source address, next hop address), we unicast the Join Reply to the next hop address.

Chapter 5

Performance Analysis

This chapter presents a performance analysis of our soft-ack techniques when applied to ODMRP. We compare the following protocols that were discussed in Chapters 2, 3: ODMRP, OMP-ps, OMP-n, OMP-np. Flooding simulations are also shown to observe how ODMRP compares with flooding.

5.1 Simulation Setup

5.1.1 Network and Mobility model

We have simulated a network of 50 nodes in a field of 1000m x 1000m. The transmission range of a node is 250m and the channel bandwidth is 2 Mb/s. The initial position of a node is setup as follows: The field is divided into squares of equal area (142m x 142m). Each node is placed in a random square chosen from the list of squares not chosen before. So, each node will be in a different square. Within the 142m x 142m square, a node is placed randomly in a 50m x 50m square at the center of the 142m x 142m square. Since there will be 49 squares, the node 50 is randomly placed in the whole field. This initial setup ensures that there are no partitions at the start of the simulation. This is helpful for simulations done with low node mobility speeds, because we can avoid partitions that generally last longer at these lower speeds. After the initial setup, nodes start moving continuously at a predefined speed throughout the simulation.

Moving directions of each node is selected randomly every 10 seconds. When a node reaches the simulation terrain boundary, it is reflected back into the terrain and continues to move.

Each run of the simulator takes a *scenario file* as input. A scenario file describes the exact motion of each node and the connectivity among nodes at all times during the simulation period. Since the performance of the routing protocols is very sensitive to the movement patterns, we have generated 5 different scenarios for each mobility speed and each simulation point is averaged over these five scenarios.

5.1.2 Traffic Pattern

We have used Constant Bit Rate (CBR) sources as our traffic sources. Data packet size is 512 bytes. Multicast sources and receivers join the multicast session at the beginning of the simulation. Hence, our simulations do not test/account for the overhead produced in the session leave process. Each simulation is run for 600 seconds to sample and gather performance statistics. We have simulated only multicast traffic. In future study, we plan to simulate combined multicast and unicast traffic.

The major parameters, used in the simulations, are shown in Table 5.1.

5.1.3 Performance Metrics

We have used the following metrics in comparing protocol performance.

- *Packet delivery ratio:* The ratio of the number of data packets actually delivered to the multicast receivers versus the number of data packets supposed to be received. This number presents the effectiveness of a protocol.

- *Number of data packets transmitted per data packet delivered:* 'data packets transmitted' is the count of every individual data transmission by each node over the entire network. This count includes data, Join Query(since data is piggybacked) and retransmissions of

Parameter	**Value**
Number of nodes	50
Field area	1000m x 1000m
Simulation duration	600 sec
Transmitter range	250 m
Channel Bandwidth	2 Mb/s
Node mobility	Random, bounce back into the field if a boundary is encountered
Speed	Fixed to a value in the range of [0,20] m/s
Traffic type	Constant Bit Rate
Data packet payload	512 bytes
Data packet header	24 bytes (IP header: 20, sequence number: 4). For *OMP-n* and *OMP-np* protocols, additional 1 byte for status code.
Join Query size	44 bytes (IP header: 20, Routing information: 24) + Data payload
Join Reply size	36 bytes (IP header: 20, Routing information: 16)

Table 5.1: Parameters used in the simulations

data packets present in our soft-ack techniques (for example, responses to NAKs, PACK retransmissions). This measure shows the efficiency in terms of channel access and is very important in ad hoc networks since link layer protocols are typically contention-based.

- *Total overhead bytes transmitted per data packet delivered):* All transmission bytes, excluding the data payload, are counted in the total overhead bytes. This includes routing packets and also bytes of data packet headers.

- *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.

5.2 Simulation Results

In this section, we describe the different types of simulations done:

- Network Traffic Load: We vary the load that we offer the network (i.e. total packet sending rate by all multicast sources), in order to study the protocols behavior at low and high loads. In our simulations, the packet sending rate at each source is same.
- Mobility Speed: We vary the node mobility speed to study how it affects the protocols' performance.
- Number of Senders: We vary the number of senders, while fixing the multicast group size.
- Multicast Group Size: We vary the number of multicast members to investigate the scalability of the protocols.

5.2.1 Network Traffic Load

To study the effect of data traffic load on the protocols, we varied the load on the network. The multicast group size is 21. There are 5 senders within this group. So, each sender has 20 receivers (same nodes send as well as receive packets). In our simulations, only members of a multicast group can be senders. Node mobility is zero in this experiment. Therefore, the packet drops are caused only by buffer overflow, collision and congestion. Since the base protocol is ODMRP, any performance difference can be attributed to congestion and collisions, due to hidden terminal problem.

Figure 5.1 gives the performances of the protocols for varying network loads. Our soft-ack protocols (i.e. OMP-n and OMP-np) have nearly identical performance in this case. Figure 5.1(a) shows the number of data packets transmitted per data packet delivered (hereafter, referred to as data transmit-delivery ratio) under different loads. Flooding has a constant data transmit-delivery ratio since it sends a packet to all nodes in the network, regardless of load. From the figure, we observe that ODMRP has the data transmit-delivery ratio almost equal to flooding. As explained in Section 3.1, ODMRP has a lot of redundant data transmissions. Our

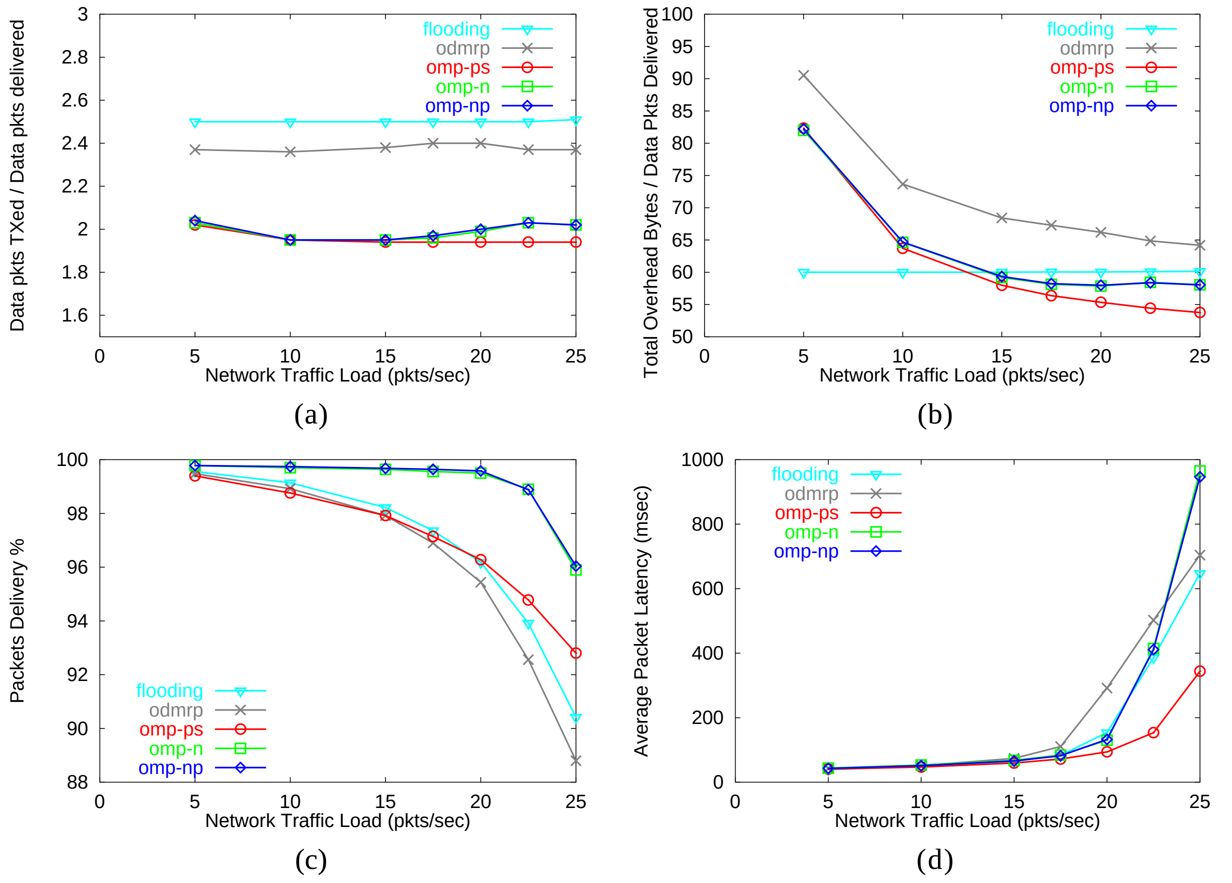

Figure 5.1: Performance of protocols as a function of Network Traffic Load. Node speed = 0 m/s, Number of senders = 5, Multicast group size = 21

techniques (OMP-ps, OMP-n, OMP-np) cut down this data transmit-delivery ratio by sending data only on the required paths. As seen in Figure 5.1, our techniques cut down the data transmit-delivery ratio by 15 - 20 %. As the network load increases, ODMRP and OMP-ps drop many packets due to collisions and congestion, that can be inferred from the drop in delivery rate in Figure 5.1(c). To recover these lost packets, OMP-n and OMP-np use the soft-ack techniques described earlier. This results in additional data transmissions and is the cause for the slight increase in the data transmit-delivery ratio when compared to OMP-ps.

Figure 5.1(b) shows the total overhead bytes transmitted per data packet delivered (hereafter, referred to as overhead-delivery ratio) under different loads. Flooding has overhead due to data headers. It is constant because each packet is transmitted by every node exactly once. Other protocols have additional overhead resulting from control packets used. Since the base protocol is ODMRP and our techniques do not add any additional control packets, the overhead-delivery ratio differences among the protocols (except flooding) is due to the data header overhead. Our techniques have lesser overhead-delivery ratio compared to ODMRP due to less number of data transmissions. As load increases, OMP-n and OMP-np have slightly higher overhead-delivery ratio than OMP-ps due to the additional recovered packets' transmissions.

Figure 5.1(c) shows the delivery rate under different loads. From the figure, we observe that ODMRP performs slightly better than OMP-ps at low to moderate loads. At high loads, however, ODMRP is worse than OMP-ps. Flooding has better performance than ODMRP at all loads. As load increases, OMP-ps has a better performance than ODMRP due to fewer data transmissions. With higher number of data transmissions, there is a greater risk of congestion and collisions, that results in ODMRP's poor performance when compared to OMP-ps. As load increases, we can observe that the performance of flooding, ODMRP and OMP-ps starts degrading. This is because of collisions due to hidden terminal problem. Using soft-acks, OMP-n and OMP-np improve OMP-ps performance and have consistently better performance compared to all the other protocols.

Figure 5.1(d) shows the average packet latency under different loads. Since our techniques reduce the number of data transmissions, congestion (that increases the average packet delay) is a less severe problem in our techniques when compared to ODMRP. As we can see from the figure, OMP-ps has the best performance. At low loads, OMP-n and OMP-np have same latency as OMP-ps. As load increases, OMP-ps starts dropping packets due to collisions and congestion. Our soft-ack techniques recover these lost packets by using the negative-acknowledgement technique. Since packets are recovered after receiving next sequence number packet, these recovered packets have an additional delay of inter-packet arrival time. At high loads, packet loss increases and increases the number of recovered packets, that in turn increases the average packet latency. Although our soft-ack techniques had a lower delay at low packet rates, the delay increased as load increases due to the recovered packets. Since our PACK technique is a congestion-based, it does not help much in reducing the latency of the NAK technique at high loads.

5.2.2 Mobility Speed

To study the effect of Mobility, we varied the mobility speed from 0 m/sec to 20 m/sec. In a given simulation, however, each node moves constantly with the predefined speed. There are 5 senders with a multicast group size of 21. The packet rate is set at 3 pkts/sec at each source (i.e. Network traffic load = 15 pkts/sec), which is a relatively low packet rate. This low packet rate is set so that we can study the effect of mobility on the protocols' performance, without creating network saturation.

Figure 5.2 shows the performances of the protocols, under different mobility speeds. The same performance observations, given in Section 5.2.1 for Network Traffic Load simulations, can also be inferred from the figure. There is a 20% reduction in the effective number of data transmissions in Figure 5.2(a) for our techniques when compared to ODMRP. Figure 5.2(b) shows that our protocols have lower overhead-delivery ratio than ODMRP due to lesser number of data transmissions. our soft-ack techniques have a higher overhead-delivery ratio than the OMP-ps protocol due to the additional recovered packets transmissions. Figure 5.2(c) shows that all protocols' delivery rate decreases as mobility speed is increased. This is due to the frequently changing network. Flooding has a slightly better performance than ODMRP in terms of delivery rate. Due to mobility, the network frequently changes, resulting in stale routes. ODMRP benefits from the redundant routes, by delivering packets on these routes. Since many of the redundant routes present in ODMRP, are not present in OMP-ps, OMP-ps's delivery rate is worse than ODMRP. But our soft-ack techniques show a better performance by recovering the packets lost due to mobility, collisions and congestion. From Figure 5.2(d),

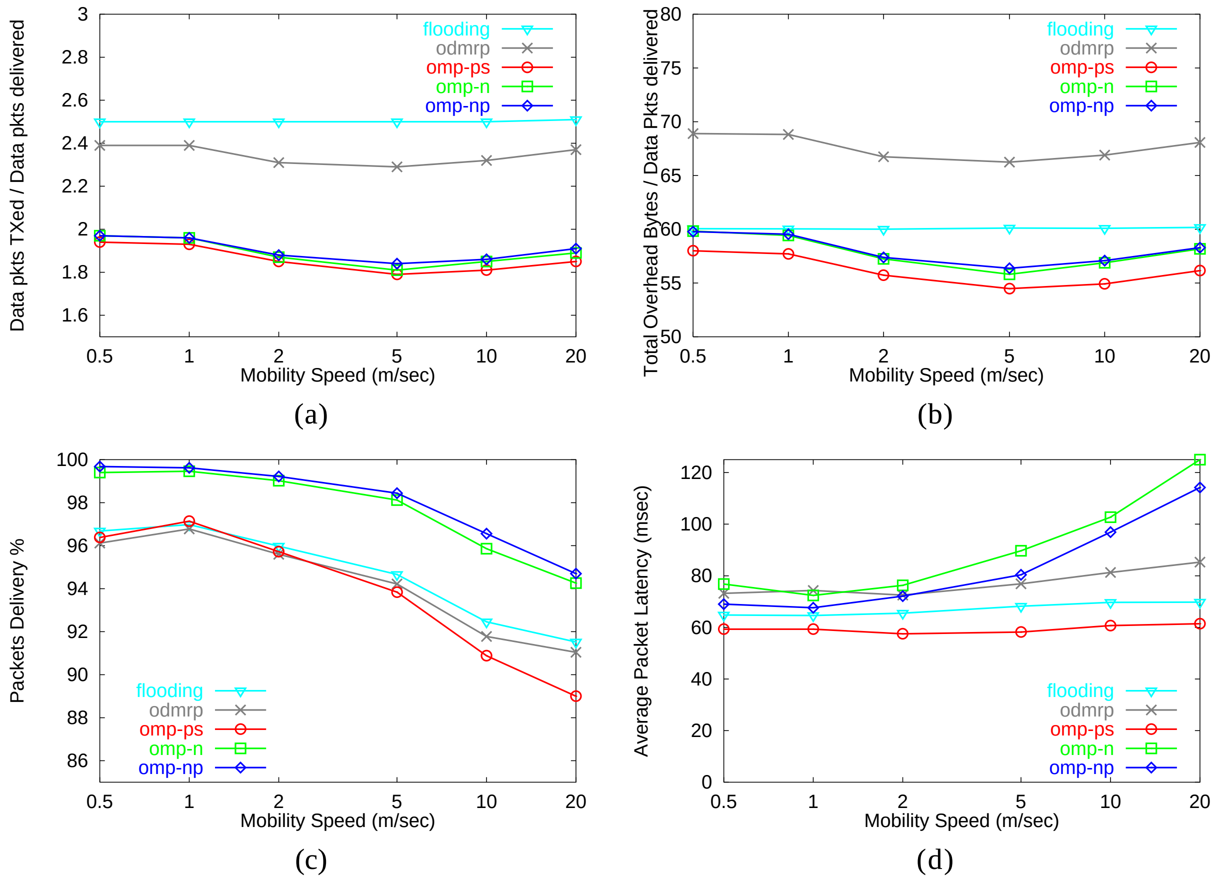

Figure 5.2: Performance of protocols as a function of Mobility Speed. Network traffic load = 15 pkts/sec, Number of senders = 5, Multicast group size = 21

we can infer the same observations made before. OMP-ps has lower average packet delay than ODMRP due to the less number of data transmissions. Our soft-ack protocols have a higher latency because they have to recover more lost packets as mobility increases and the increasing recovered packets contribute towards more additional delay, thereby increasing the average latency. Our PACK technique (OMP-np) reduces the number of NAK responses (note that these packets have an additional inter-packet-arrival time delay and are the cause for the high delay of our NAK technique), thereby decreasing the average latency and slightly better delivery rate (observed in Figure 5.2(c)) than the NAK (OMP-n) technique.

5.2.3 Number of Senders

In this experiment, we set the multicast group size to be constant at 21. Node speed is set at a constant 1 m/sec and network traffic load set to 15 pkts/sec. These values are relatively low and help in studying the effect of Number of Senders on our techniques. The Number of Senders is varied from 1 to 20.

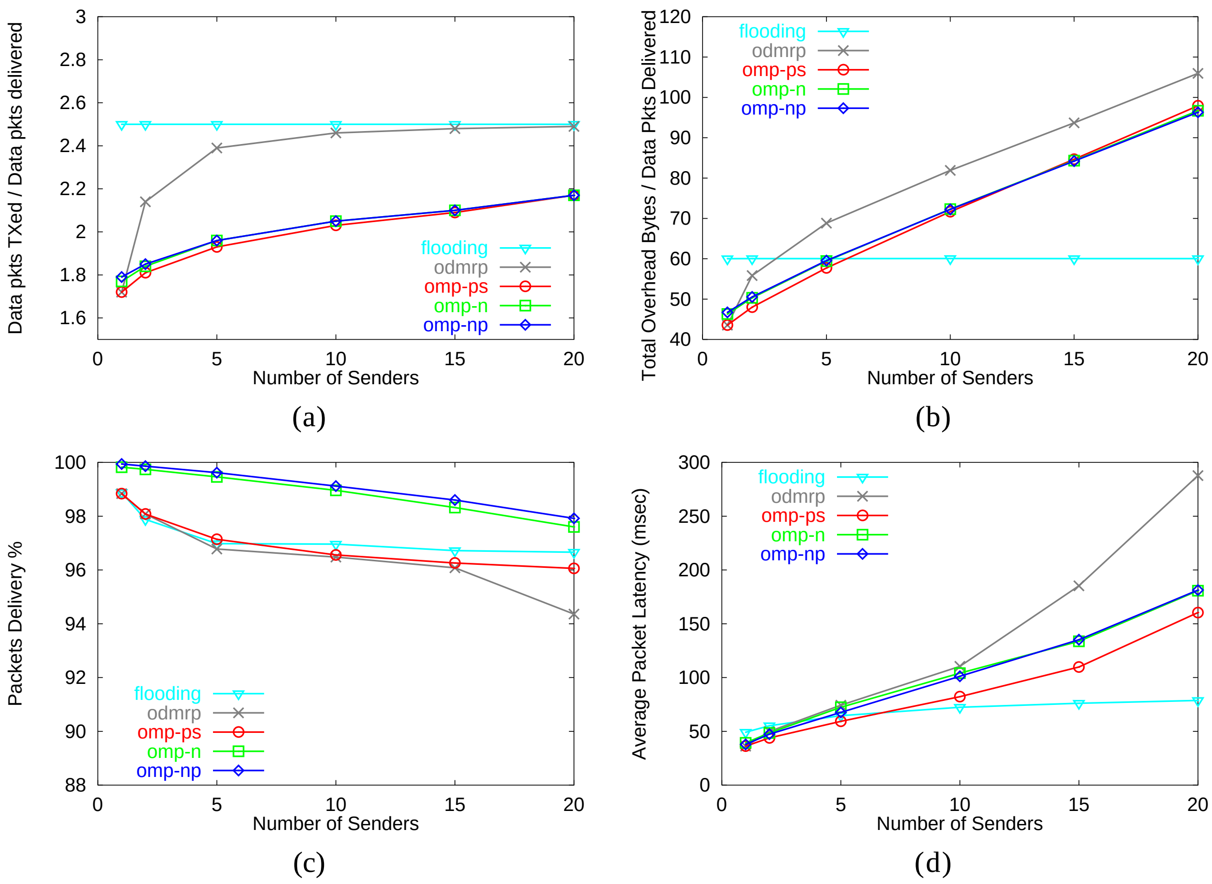

Figure 5.3: Performance of protocols as a function of Number of Senders. Node speed = 1 m/s, Multicast group size = 21, Network traffic load = 15 pkts/sec.

Figure 5.3 shows the performance of the protocols with varying Number of Senders. From Figure 5.3(a) we observe following: when the Number of Senders is 1, ODMRP and OMP-ps techniques have the same Forwarding Group (FG) mesh. As the number of senders increase, the number of data transmissions increase due to more number of nodes in FG mesh. This can be be observed in all the protocols using ODMRP as base protocol. Since ODMRP uses the whole mesh and OMP-ps uses only the per-sender FG mesh, ODMRP's number of data transmissions is very high compared to OMP-ps. We can also observe that ODMRP's number of data transmissions become almost equal to that of flooding, that indicates that ODMRP is using the entire network as its mesh members. Same performance observations, given in Section 5.2.1 for network traffic load simulations, for overhead-delivery ratio, packet delivery rate and delay can be inferred from Figure 5.3 (b),(c),(d).

5.2.4 Multicast Group Size

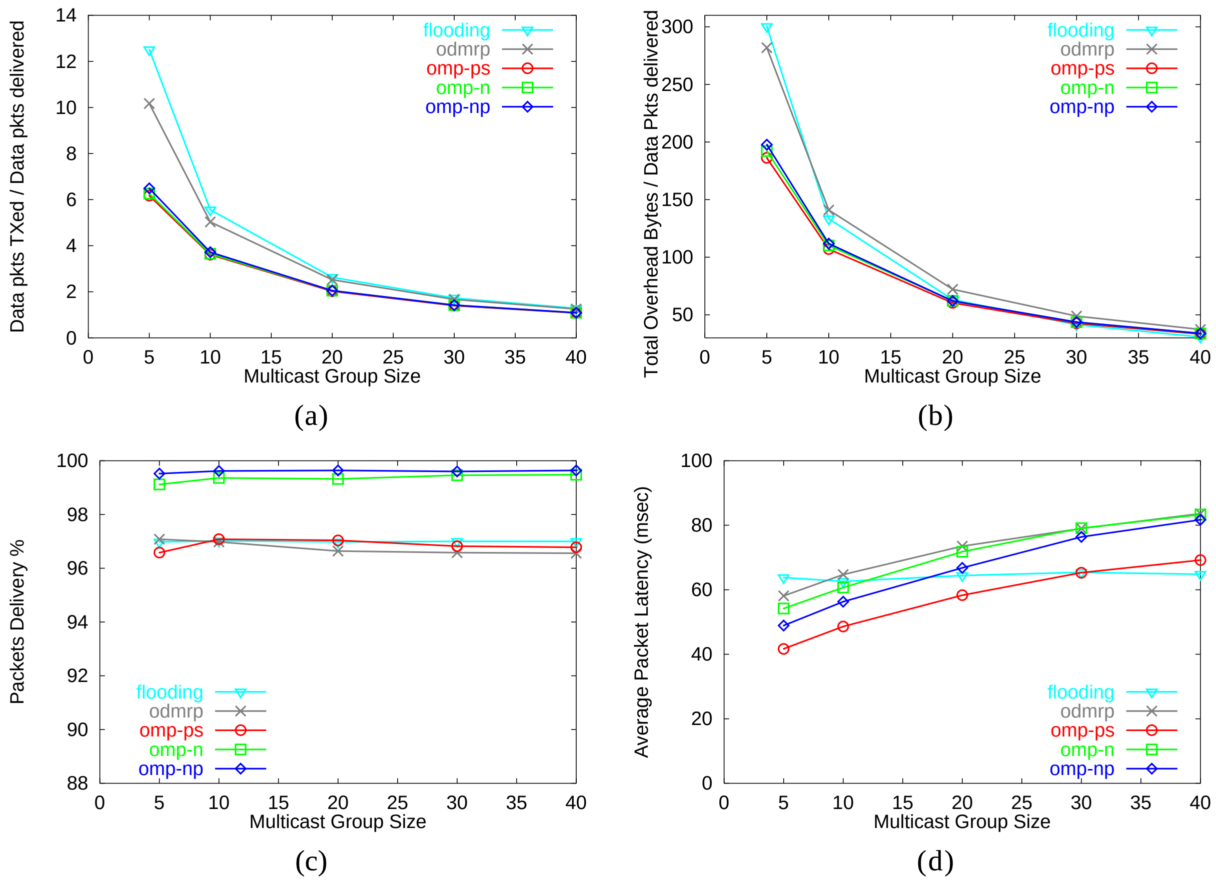

Figure 5.4: Performance of protocols as a function of Multicast Group size. Node speed = 1 m/s, Number of senders = 5, Network traffic load = 15 pkts/sec

We vary the Multicast Group Size to investigate the scalability of the different protocols. The number of senders is fixed at 5. Node speed is set at a constant 1 m/sec and network traffic load at 15 pkts / sec. Group size is varied from 5 to 40. Figure 5.4 shows the performance of the protocols for varying Multicast group size.

From Figure 5.4(a), we observe that the number of data transmissions reduction by our techniques is between 15% to 35%. For packet delivery rate, average packet latency and total overhead, performance observations are same as that made in Section 5.2.1. Since the packet rate is low, OMP-np reduced the average latency by decreasing the number of NAK responses. This can be observed in the Figure 5.4(d).

Chapter 6

Conclusions

The hidden-terminal issue is a major problem for broadcasts and multicasts. It can cause packet collisions and decrease the one-hop broadcast success rate, thereby significantly decreasing the performance of multicast routing protocols. To overcome the unreliable one-hop broadcasts problem, mesh-based multicast schemes [15, 8] use redundant paths to receivers to ensure high delivery rate. But this is not efficient, especially at high loads and in congested networks.

To improve the reliability of one-hop broadcasts, we describe two forms of soft acknowledgement (soft-ack) schemes. In the first scheme, called Negative-acknowledgement (NAK) scheme, forwarding nodes or receivers of a multicast group advertise any missing packet numbers. To reduce the overhead, the NAK information is piggybacked on the data packets as much as possible. Based on this information, neighbors respond by retransmitting any missing data packets. The NAK scheme significantly improves the throughput of a protocol. But it also increases the average packet delay because all the NAK response data packets have an additional delay of at least one inter-packet arrival interval. To reduce the average delay, we use the second scheme, called Passive-acknowledgement (PACK) scheme, in addition to the NAK scheme. In the PACK scheme, a node forwarding a packet expects to hear another transmission (possibly retransmission by one of its child nodes) of the packet. If it does not hear a retransmission within certain time, the node will retransmit the packet again. To reduce excessive retransmissions, we use channel congestion and perceived packet rates to determine when to retransmit a packet.

The proposed soft-ack schemes can be applied in general to any MANET communication protocol that requires highly reliable one-hop broadcasts. Typical applications include data transmissions in multicast protocols, and control or routing packet broadcasts in unicast and multicast protocols.

To demonstrate the benefits of our soft-ack schemes, we have applied them to ODMRP [15], a recently proposed multicast algorithm for MANETs. ODMRP, being a mesh-based protocol, has a large number of redundant packet transmissions. We incorporate a small modification to ODMRP that cuts down the number of redundant packet transmissions. Compared to the original ODMRP, this modification improves packet latencies and sometimes reduces the throughput since redundant transmissions are cut down. To improve any loss in throughput, we have fortified the modified ODMRP with the NAK and PACK schemes. The modified ODMRP with NAK is called *OMP-n* and the modified ODMRP with both NAK and PACK is called *OMP-np*.

We have implemented the original ODMRP, OMP-n and OMP-np using the Network Simulator 2 (*ns-2*) with mobility extensions from CMU [6, 4]. We have tested all protocols, using simulations of a network of 50 nodes moving randomly in a square field of 1000m x 1000m. We have tested our schemes under different network scenarios by varying traffic loads, node speeds, number of data packet senders and different multicast group sizes.

Our simulations indicate that the soft-ack schemes improve the delivery rate and number of packet transmissions per delivered packet significantly, without increasing packet latencies significantly. In particular, our soft-ack protocols reduce the number of data packet transmissions per delivered packet by 15-35 %, compared to the original ODMRP. Furthermore, OMP-n and OMP-np offer superior delivery rates under all cases. OMP-n and OMP-np have similar delivery rates, while OMP-np has lower packet latencies than OMP-np.

In future, we plan to test our schemes by varying different simulation parameters like the field size, the number of nodes and the node movement pattern. We would also like to apply our schemes to other protocols, in particular to tree-based protocols like AODV. Since tree-based protocols do not have redundant data transmissions like a mesh-based protocol, we do not expect much reduction in packet transmissions but expect significant delivery rate improvement. It will be interesting to investigate the benefits of our soft-ack schemes for unicast routing algorithms, especially the ones that broadcast route request packets to discover routes.

Bibliography

[1] G. Aggelou and R. Tafazolli, “Rdmar: A bandwidth-efficient routing protocol for mobile ad hoc networks,” in *WoWMoM*, Aug. 1999.

[2] A. Ballardie, “Core based trees (CBT) multicast routing architecture.” Internet Draft, 1997.

[3] E. Bommaiah, A. McAuley, R. Talpade, and M.-K. Liu, “Ad hoc multicast routing protocol.” IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-talpade-manet-amroute-00.txt, 1998.

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

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

[6] 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.

[7] D. Farinacci, A. Thaler, S. Handley, C. Liu, and L. Wei, “Protocol independent multicast-sparse mode (PIM-SM).” RFC 2117, June 1997.

[8] J. J. Garcia-Luna-Aceves and E. L. Madruga, “The core-assisted mesh protocol,” in *IEEE Journal on Selected Areas in Communications,vol. 17,no. 8,* pp. 1380–1394, Aug. 1999.

[9] J. J. Garcia-Luna-Aceves and E. L. Madruga, “A multicast routing protocol for ad-hoc networks,” in *IEEE INFOCOM’99*, Mar. 1999.

[10] 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.

[11] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, “Flooding for reliable multicast in multi-hop ad hoc networks,” in *ACM Dialm’99*, Aug. 1999.

[12] 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.

[13] P. Jacquet, P. Muhlethaler, and A. Qayyum, “Optimized link state routing protocol.” IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-01.txt, 2000.

[14] 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.

[15] S.-J. Lee, W. Su, and M. Gerla, "On-demand multicast routing protocol for ad hoc networks." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-odmrp-02.txt, 2000.

[16] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A performance comparison study of ad hoc wireless multicast protocols," in *IEEE Infocom 2000*, 2000.

[17] B. N. Levine and J. J. Garcia-Luna-Aceves, "A comparison of known classes of reliable multicast protocols," in *Multimedia Systems (ACM/Springer), Vol. 6, No.5*, Aug. 1998.

[18] J. Meggers and G. Filios, "Multicast communication in ad hoc networks," in *VTC'98*, Apr. 1998.

[19] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *IEEE/ACM MOBICOM'99*, pp. 151–162, 1999.

[20] R. G. Ogier, "Efficient routing protocols for packet-radio networks based on tree sharing," in *MOMUC'99*, Nov. 1999.

[21] 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.

[22] W. Peng, "Efficient broadcast in mobile ad hoc networks using connected dominating sets," in *ICPADS*, 2000.

[23] 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.

[24] C. E. Perkins, "Ad hoc on demand distance vector routing." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-02.txt, 1997.

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

[26] T. Pusateri, "Distance vector multicast routing protocol." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-idmr-dvmrp-v3-07.txt, 1998.

[27] S. Ramakrishnan and B. N. Jain, "A negative acknowledgement with periodic polling protocol for multicast over lan," in *IEEE INFOCOM*, pp. 502–511, Mar. 1987.

[28] R. Sivakumar et al., "Core extraction distributed ad hoc routing (CEDAR) specification." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-cedar-spec-00.txt, 1997.

[29] W. Su and S.-J. Lee, " ." Personal communication, 1999.

[30] F. A. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part ii - the hidden terminal problem in carrier sense multiple-access modes and the busy-tone solution," in *IEEE Trans. Commun.,vol. COM-23,no. 12*, pp. 1417–1433, 1975.

[31] C. W. Wu, Y. C. Tay, and C.-K. Toh, "Ad hoc multicast routing protocol utilizing increasing id-numbers (AMRIS)." IETF Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-manet-amris-spec-00.txt, 1998.

Vita

Shanmukha Rao Voona was born in Sompeta, India on November 28, 1975, the son of Padmavati Voona and Ramanandam Voona. After completing his schooling at Vidya Vihar, Vishakhapatnam, he entered the Vignan Co-operative Junior College, Guntur in 1991 for his college studies. He received the degree of Bachelor of Technology in Computer Science from Regional Engineering College, Warangal in May 1997. He worked at Wipro GE Medical Systems Ltd., Bangalore for one year and in August 1998, he entered the graduate program in Computer Science at The University of Texas at San Antonio, USA.