A New Wireless Ad hoc Multicast Routing Protocol

Seungjoon Lee, Chongkwon Kim
School of Computer Science and Engineering
Seoul National University
Seoul, Korea
{leesj,ckim}@popeye.snu.ac.kr

June 29, 2001

Abstract

An ad hoc network is a multi-hop wireless network of mobile nodes without the intervention of fixed infrastructure. Limited bandwidth and mobility require that ad hoc routing protocols be robust, simple, and energy-conserving. This paper proposes a new ad hoc multicast routing protocol called Neighbor-Supporting Multicast Protocol (NSMP). NSMP adopts a mesh structure to enhance resilience against mobility. And NSMP utilizes node locality to reduce the overhead of route maintenance. NSMP also attempts to improve route efficiency and reduce data transmissions. Our simulation results show that NSMP delivers packets efficiently while substantially reducing control overhead in various environments.

1 Introduction

An ad hoc network is a multi-hop wireless network formed by a collection of mobile nodes without the intervention of fixed infrastructure. Because an ad hoc network is infrastructure-less and self-organized, it is used to provide impromptu communication facilities in inhospitable environments. Typical application areas include battlefields, emergency search and rescue sites, and data acquisition in remote areas. An ad hoc network is also useful in classrooms and conventions where participants share information dynamically through their mobile computing devices.

Each mobile node in an ad hoc network functions as a router to establish end-to-end connections between any two nodes. Although a packet reaches all neighbors within transmission range, a mobile node has limited transmission ranges and its signals may not reach all hosts. To provide communications throughout the network, a sequence of neighbor nodes from a source to a destination form a path and intermediate mobile hosts relay packets in a store-and-forward mode.

Unique characteristics of an ad hoc network raise several requirements for the routing protocol design: ad hoc network routing must be simple, robust and minimize control message exchanges. Ad hoc routing must be simple because routing is performed by generic mobile hosts which have limited CPU and memory capacities and are powered by batteries. Bandwidth is a scarce resource in wireless networks. Routing algorithms which consume excessive bandwidth for routing control message exchanges may not be appropriate for wireless networks. The topology of an ad hoc network is inherently volatile and routing algorithms must be robust against frequent topology changes caused by host movements.

Many routing schemes have been presented to provide adequate performance of ad hoc networks. Ad hoc routing is classified into proactive routing and reactive routing based on when routes are determined. Proactive routing continuously makes routing decisions so that routes are immediately available when packets need to be transmitted. DBF[1], DSDV[2], WRP[3] are proactive routing protocols. Reactive routing determines routes on an as-needed basis: when a node has a packet to transmit, it queries the network for a route. TORA[4], DSR[5], AODV[6], ABR[7], RDMAR[8] belong to reactive routing. Proactive routing consumes a great deal of radio resources to exchange routing information. Also, pre-determined routes may rapidly lose their validity in an ad hoc network because its topology changes rapidly. Previous study showed that reactive protocols performed better than proactive protocols [9], [10], [11].

In addition to unicast routing protocols, several multicast routing protocols for ad hoc networks have been proposed in more recent years [12], [13], [14], [15], [16]. Unicast is a special form of multicast, and some proposed multicast routing protocols support both unicast and multicast routing [12], [13]. Proposed multicast routing can be classified into tree-based protocols and mesh-based protocols. Tree-based protocols [12], [15], [16] build a tree that connects all multicast members. Tree-based protocols are generally more efficient than mesh-based protocols, but they are not robust against topology changes because there is no alternative path between a source and a destination. As a result, every link failure in a multicast tree may involve a set of control message exchanges for tree re-build. Contrary to tree-based protocols where there is only one path between any two nodes, mesh-based protocols allow redundant paths between two nodes because mesh-based multicast protocols provide alternative paths and a link failure need not trigger the recomputation of a mesh. Previous studies showed that mesh-based protocols are robust against topology change and are more suitable than tree-based protocols [14], [17].

Although reactive routing protocol finds paths on demand and uses "soft state" to avoid stale route information, route failure still occurs due to frequent topology changes in ad hoc networks. Most ad hoc routing protocols have path maintenance mechanisms that provide adequate connectivity under topology changes. We can reduce the path maintenance cost using old path information and node locality information. One method to reduce the path maintenance cost is to use an old path segment and search a new path segment starting from the link failure point. "Expanded Ring Search (ERS)" is typically used for the new path segment search to further reduce the path maintenance cost. More recently, more systematized efforts for localized repair have been proposed in the context of unicast routing [8] [18].

ODMRP is an ad hoc multicast routing protocol based on a multicast mesh [13]. In ODMRP, if a source node has data to send, it periodically broadcasts "Join Request" to find and maintain multicast routes. All the other nodes re-broadcast the packet when they receive non-duplicate one. When a multicast group member receives "Join Request", the node replies with "Join Table." And subsequent replies by the nodes along a reverse path establish a route. ODMRP uses soft states, so leaving a group is automatically handled by timeout. As shown, ODMRP relies on frequent network-wide flooding, which may lead to a scalability problem when the number of source nodes is large. The control packet overhead becomes more prominent when the multicast group is small in comparison with the entire network.

In this paper, we present a new on-demand multicast routing protocol called *Neighbor-Supporting Multicast Protocol (NSMP)*. NSMP is a robust, low overhead and efficient protocol. We choose to use the mesh infrastructure because resilience against link failures is an important property of ad hoc multicast routing. NSMP is the first multicast routing protocol that exploits node locality for route maintenace except for basic ERS. Broadcasts are expensive operations in ad hoc networks [19]. NSMP minimizes the frequency of control message broadcasts. Broadcasts are occasionally used for initial route establishment or a network partition repair. For normal and periodic mesh maintenances, control messages reach only forwarding nodes and their neighbor nodes. In selecting a new route, NSMP prefers a path that contains existing forwarding nodes. Thus, NSMP enhances the route efficiency by reducing the number of forwarding nodes.

We have evaluated the performance of NSMP via computer simulation. The simulation result shows that NSMP is robust against frequent toplogy changes and delivers more than 96% of data packets under reasonable simulation environments. Moreover, data packet transmissions and control message exchanges are reduced by 5-30% compared to existing mesh-based ad hoc multicast routing protocols.

The rest of this paper is organized as follows. Section 2 contains an overview of NSMP, and a more detailed description of NSMP is presented in section 3. Section 4 provides results of simulation experiments, and section 5 concludes the paper.

2 A New Multicast Routing Protocol

2.1 An Overview of NSMP

NSMP is a robust, yet efficient ad hoc multicast routing protocol. Mesh infrastructure used in NSMP has resilience against link failures. A soft state approach is used, and routes are built and maintained with basic route discovery and reply messages. NSMP also operates independent of unicast routing protocol.

Localizing route discovery and maintenance operations, NSMP reduces the routing overhead. As discovered in RDMAR [8], most link failure recoveries can be localized to a small region along a previous route. NSMP performs two types of route discovery: *flooding route discovery* and *local route discovery*. For routine path maintenances, NSMP uses local route discovery which is restricted only to a small set of mobile nodes directly related to a multicast group. For an initial route establishment or a network partition repair, NSMP occasionally performs flooding route discovery in which control messages are broadcast by all nodes. For long-lived connections, routine path maintenances occur many times more frequently than the initial path establishment, and the saving by localized path maintenance could be sizable.

NSMP attempts to achieve the route efficiency of the multicast tree while enjoying the robustness of the multicast

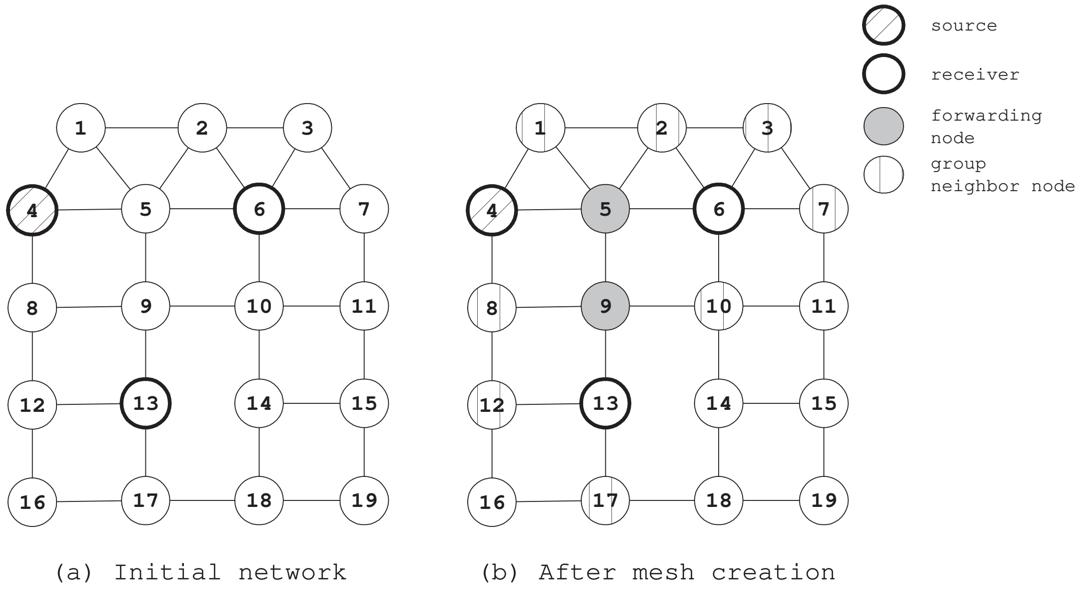

Figure 1: Multicast mesh creation

mesh infrastructure. It is known that the mesh structure is more robust against topology changes than the tree structure [14], [17]. However, previous study [17] found that tree-based protocol transmits less data packets than mesh-based protocol. In selecting a route, NSMP prefers a path that contains existing forwarding nodes to reduce the number of forwarding nodes. This enhances route efficiency, leading to less contention in the network.

2.2 Multicast Mesh Creation

A new source initially sends a FLOOD_REQ packet. The FLOOD_REQ packet has an upstream node field. When an intermediate node receives the FLOOD_REQ packet, it caches the upstream node and updates the field with its own address before forwarding it to next nodes. When a receiver receives the FLOOD_REQ packet, it sends a REP packet to the node from which it received the packet. The upstream node receives the REP packet and adds an entry for the group to its routing table. Then it forwards the REP packet to its own upstream node, and the REP packet eventually reaches the source node. The intermediate nodes that relay the REP packet become *forwarding nodes*. A multicast mesh of a group consists of sources, receivers, forwarding nodes, and links connecting them. The nodes in a multicast mesh are called *mesh nodes*.

Figure 1 illustrates how a multicast mesh is built. Assume that nodes 6 and 13 are receivers of a multicast group. When node 4 joins the group as a source, it broadcasts a FLOOD_REQ packet. Node 5 receives the packet and broadcasts it. When node 6 receives the FLOOD_REQ packet, it sends a REP packet to its upstream, node 5. When node 5 receives the REP packet, it knows that it is on the multicast mesh and relays the packet to its upstream, node 4. Similarly, node 13 also sends a REP packet when it receives a FLOOD_REQ. Although there are other

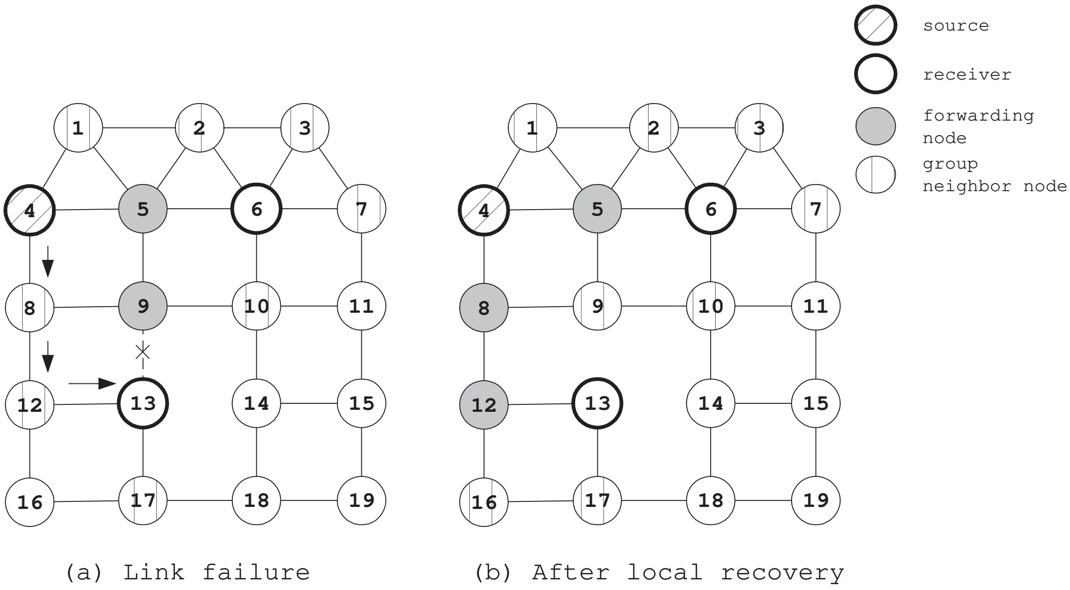

Figure 2: Multicast mesh maintenance

reverse paths for this REP to take (e.g. 13-12-8-4), we assume here that REP takes the path (13-9-5-4) and that node 9 becomes a forwarding node. Figure 1 (b) shows the resulting multicast mesh. When a source transmits a DATA packet, only forwarding nodes relay the packet, so that the packet is delivered to receivers along an established mesh.

Now let us consider *neighbor nodes* of the multicast mesh. Neighbor nodes are nodes that are directly connected to at least one mesh node. In Figure 1 (b), nodes 1, 2, 3, 7, 8, 10, 12, and 17 are the neighbor nodes. Forwarding nodes and group neighbor nodes lose their function unless they are refreshed within pre-defined timeout period. Section 3 shows detailed procedures of how a multicast mesh is built and a node becomes a group neighbor.

2.3 Multicast Mesh Maintenance

2.3.1 Local Route Discovery

Each source periodically transmits a LOCAL_REQ packet, and only mesh nodes and group neighbor nodes relay the packet. Therefore, all nodes two hops away from the mesh nodes receive the LOCAL_REQ packet. This mechanism can reduce control overhead, and due to node locality, it repairs most link failures caused by node movements. REP packets to LOCAL_REQ packets are relayed to a source in the same way as REP packets to FLOOD_REQ packets in section 2.2. Forwarding nodes and group neighbor nodes along a multicast mesh are updated as REP packets are relayed to a source.

For example, assume that a failure occurs to a link (9, 13) in Figure 2. Node 4 will eventually send a LOCAL_REQ packet since each source periodically performs local route discovery. When node 8 receives the packet, it broadcasts

Type	Sequence Number	Group Address	Source Address	Upstream	FC	NC

Figure 3: Packet header of NSMP

the packet since group neighbor nodes relay LOCAL_REQ packets. When node 12 subsequently broadcasts the packet, node 13 receives it and sends a REP packet to build a new route to the source. The repaired mesh is shown in Figure 2 (b). Note that more than 30% of the nodes (i.e. six nodes) in Figure 2 (a) do not re-broadcast the LOCAL_REQ packet.

Local route discovery ensures lower control overhead, but it does not repair all link failures. Suppose that a link (8, 12) in Figure 2 (b) failed. Local route discovery cannot repair this link failure. With reasonable network connectivity, however, *locally unrecoverable* link failures occur less frequently than link failures that can be repaired by local route discovery. Simulation results in section 4 show that local route discovery is effective under various environments.

2.3.2 Flooding Route Discovery

NSMP uses flooding route discovery in several cases. When a node becomes a new source, it sends a FLOOD_REQ packet in order to create an initial mesh. In NSMP, a node within two hops away from mesh nodes can join the group as a receiver by replying to a LOCAL_REQ packet. However, a node more than two hops away from the mesh nodes must flood a MEM_REQ packet. In addition, network partitions only can be recovered by FLOOD_REQ packets.

2.4 Route Efficiency Improvement

In selecting a route, NSMP gives a preference to a path that contains more existing forwarding nodes. The level of preference is an important parameter that trade-offs the routing efficiency and path robustness. Assume that node 17 becomes a new receiver in Figure 1 (b). And further assume that node 17 receives two route discovery packets: one from the path (4, 5, 9, 13, 17) and the other from the path (4, 8, 12, 16, 17). Both paths have the same length. However, the path (4, 5, 9, 13, 17) uses the existing path and the path (4, 8, 12, 16, 17) requires three new forwarding nodes. In terms of route efficiency, the former is better than the latter, and vice versa in terms of robustness.

3 Detailed Description of NSMP

3.1 Data Structures and Packet Header

Figure 3 shows the packet header of NSMP. We also assume the availablity of *ttl* field in other protocol (e.g. IP) used together. *Forward Count (FC)* denotes the number of forwarding nodes along a path. A forwarding node increases

Group Address	Forwarding Flag	Forwarding timeout	GroupNeighbor Flag	GroupNeighbor timeout
.....	.....	.....	.....	.....

Figure 4: Routing table used in NSMP

the *FC* by one before relaying a route discovery packet. *Non-forward Count (NC)* is the number of non-forwarding nodes. *Type* field is one of the following values:

- DATA: data packet
- FLOOD_REQ: flooding route discovery packet sent by a group leader
- LOCAL_REQ: local route discovery packet sent by a source
- MEM_REQ: route discovery packet sent by a new receiver
- REP: reply packet to a route discovery packet

Every node maintains a *routing table.* Figure 4 shows the fields of an entry in a routing table. When a node becomes a forwarding node of a group, it sets corresponding *ForwardingFlag.* It sets *GroupNeighborFlag* when it becomes a group neighbor node. *Forwarding timeout* and *GroupNeighbor timeout* fields denote the times when a node loses its function.

In addition, every node maintains a *DataCache* and a *ReqCache* to detect duplicate data packets and route discovery packets, respectively. The structures of the two caches are shown in Figure 5. Every source node needs to maintain a *SourceList* that consists of source addresses of the same group.

3.2 Initiating and Relaying FLOOD_REQ and LOCAL_REQ

When a node becomes a multicast source, it transmits an initial FLOOD_REQ packet. After that, all sources periodically transmit LOCAL_REQ packets at every REQ_PERIOD interval. REQ_PERIOD is important to the performance of NSMP and should be carefully adjusted according to network environments. As briefly discussed in section 2.3, NSMP uses flooding to recover network partitions. For this purpose, a group leader is selected among sources. The group leader sends FLOOD_REQ packets at every FLOOD_PERIOD interval. *Upstream* and *Source Address* fields are set to its own address, and *FC* and *NC* are set to zero.

When a node receives a route discovery packet, it consults *ReqCache* to find whether the packet has a more recent sequence number. (Group Address, Source Address, Sequence Number) fields in *ReqCache* are used to determine if

Source Address	Group Address	Sequence Number
.....	.....	.....

(a) DataCache

Source Address	Group Address	Sequence Number	Upstream
.....	.....	.....	.....

(b) ReqCache

Figure 5: Caches used in NSMP

the packet is duplicate. If the packet is a new one, the receiving node updates the corresponding entry of *ReqCache* to have correct information about *Sequence Number* and *Upstream*.

A node relays all FLOOD_REQ packets. However, it relays LOCAL_REQ packets only if it is either a mesh node or a neighbor node of the group. Before relaying a route discovery packet, a node must change *Upstream* field with its own address for later reverse path establishment. A relaying node increments *FC* by one if it is a forwarding node; otherwise, *NC* is incremented by one. Handling duplicate route discovery packets is described in section 3.3.

3.3 Initiating and Relaying REP

A path from a source to a receiver is established when a REP packet is forwarded along the reverse path from the receiver to the source. The reverse path is already recorded in the *Upstream* field of the *ReqCache*. When an intermediate node receives the REP packet, it sets the *ForwardingFlag* bit and refreshes the *Forwarding timeout* of its routing table. Then the intermediate node relays the REP packet to its upstream node. Note that a packet is broadcast to all neighbor nodes in wireless network. All nodes that detect the REP packet (except mesh nodes) become neighbor nodes of the group. The neighbor nodes set the *GroupNeighborFlag* and refresh the *GroupNeighbor timeout* of its routing table.

As explained before, NSMP tries to balance the routing efficiency and path robustness, giving preference to paths that contain more forwarding nodes. A receiver receives many route discovery packets. When a receiver receives a first non-duplicate route discovery packet, it stores the information of the packet header into *ReqCache* and delays sending REP for a short time. It computes the weighted path length, using different weights for *FC* and *NC*. The choice of these weights is also important in achieving route efficiency and resilience againt mobility. If the receiver receives another route discovery packet within the waiting period, it computes the weighted path length. If the new path is better than the currently best path, then the receiver replaces the *ReqCache* with the information of the new path. It sends a REP packet using the information of the best path stored in *ReqCache* after pre-determined time elapses since the non-duplicate route discovery packet reception. When an intermediate node receives a duplicate route discovery packet, it also checks if the new weighted path length is better than the one in *ReqCache*. If so, it stores the better upstream node in *ReqCache*. This information is not further relayed but used later if it receives a REP packet and needs to relay it to its upstream node.

NSMP ensures partition recovery by performing flooding route discovery. When previously disconnected partitions have regained connectivity, a FLOOD_REQ packet from one partition will eventually reach a receiver in another partition. Partition is recovered when a REP packet is sent and relayed across previous partitions. Larger FLOOD_PERIOD may introduce longer delay in partition recovery, so flooding route discovery needs to be performed more often in case of lower network connectivity.

3.4 Becoming a Group Neighbor

In previous subsection, we already explained the procedure of when a node becomes a neighbor node of a multicast group. Another case to become a group neighbor is when a non-mesh node finds that one of its neighbors is a source. If *Upstream* field of a route discovery packet is the same as *Source Address* field of the packet, the node becomes a group neighbor. Table 1 summarizes node behaviors when it receives route discovery packets.

3.5 Receiving and Forwarding DATA Packets

When a node receives a DATA packet, it consults *DataCache* to see if the packet is duplicate. If so, it discards the packet. Otherwise, it updates *DataCache* to reflect the packet header information, especially the sequence number. And the packet is re-broadcast if the receiving node is a forwarding node.

3.6 Joining and Leaving a Group

When a node wants to join a group as a receiver, it waits for a LOCAL_REQ packet for REQ_PERIOD. It will receive one and be able to build a route if it is a mesh node, a neighbor node of the group, or two hops away from the mesh. For example, nodes 11, 14, and 18 in Figure 2 (b) will receive a LOCAL_REQ packet within REQ_PERIOD.

If the new receiver does not receive a LOCAL_REQ packet, it broadcasts a MEM_REQ packet. On receiving a MEM_REQ packet, a node operates analogous when it receives a FLOOD_REQ packet; it needs to update an entry in *ReqCache*. MEM_REQ uses a *ttl* field. All nodes that receive a MEM_REQ packet relay the packet only if *ttl* value

Route Discovery		source	receiver	forwarding node	group neighbor	other node
Flooding		update SourceTable	send REP		*	**
		Relay	Relay	Relay	Relay	Relay
Local		update SourceTable	send REP		*	**
		Relay	Relay	Relay	Relay	

* GroupNeighbor timeout is refreshed if Source Address = Upstream

** The node becomes a group neighbor if Source Address = Upstream

Table 1: Summary of node behaviors when a route discovery packet arrives

is greater than zero. *ttl* value is decremented by one whenever it is relayed.

Source nodes and forwarding nodes send a REP packet when they receive a MEM_REQ packet. REP packets to MEM_REQ packets are relayed toward the new receiver in the same way as REP packets to FLOOD_REQ or LOCAL_REQ. The reception of a REP packet to a MEM_REQ packet also requires routing table update. And some nodes become forwarding nodes or neighbor nodes according to *Upstream* field of the REP packet.

NSMP uses expanded ring search (ERS) to reduce the bandwidth usage for MEM_REQ packets. The value of *ttl* field in the initial MEM_REQ is set to three because the new receiver sending MEM_REQ (for example, node 15 or 19 in Figure 2 (b)) is more than two hops away from the mesh. If the new receiver fails to receive any REP packet within timeout, then it floods a MEM_REQ packet.

Multiple REP packets to a MEM_REQ packet may increase the number of forwarding nodes more than necessary. This problem, however, will be resolved by timeout since only one path will be refreshed when the receiver receives route discovery packets and replies to them.

Leaving a group in NSMP does not need any additional control messages. When a node leaves a group, it does not send REP packets to subsequent route discovery packets, and soft states stored in intermediate nodes will expire.

3.7 Electing a Group Leader

A group leader is the source node whose address is the smallest among source nodes in the same multicast group. Since every source periodically sends a LOCAL_REQ packet, a source can have up-to-date information about other sources. When a source receives a LOCAL_REQ packet from another source of the same group, it updates *SourceList* to include the source address. With this information, a source can determine if its address is the smallest or not.

An entry in a *SourceList* is deleted if no LOCAL_REQ packets from the corresponding source are received within pre-determined time, for example, two times REQ_PERIOD.

4 Performance Simulation

4.1 Simulation Environment

ns-2 simulator was used for performance simulation. *ns*-2 is originally developed by the University of California at Berkeley and the VINT project [20] and recently extended to provide simulation support for ad hoc networks by the MONARCH project [21] at Carnegie Mellon University. [9] gives a detailed description about physical layer, data link layer, and IEEE 802.11 MAC protocol used in the simulation.

The protocol simulation consists of 50 wireless mobile nodes forming an ad hoc network, moving around over a square (1000 meters $\times$ 1000 meters) flat space for 300 seconds of simulated time. Nodes move according to the "random waypoint" model [9] without pause time. A multicast source generates 128-byte data packets with constant bit rate (CBR) of eight packets per second.

A number of *movement scenario files* and *group scenario files* were generated and used as inputs to the simulations. Each movement scenario file records movements of 50 mobile nodes, and the speeds of mobile nodes are uniformly distributed up to a maximum speed. Group scenario files describe receiver and source nodes and also describe the time when they join and leave a group. Each group scenario file has one multicast group. However, in one experiment, we use three group scenarios on average to simulate the multiple group environments.

In comparing the protocols, the following metrics were used.

- **Data Packet Delivery Ratio**: The percentage of data packets correctly delivered to multicast receivers
- **Number of Data Transmissions per Data Packet Delivered**: This metric reveals how efficiently a protocol establishes a path.
- **Number of Control Packets per Data Packet Delivered**: This metric represents control overhead of each protocol.

We compare the performance of NSMP with those of ODMRP and MAODV (Multicast AODV) [12]. It was assumed that no nodes were equipped with GPS, so a source in ODMRP periodically flooded "Join Request" packets. We found that "Join Reqeust" period and REQ_PERIOD of two seconds showed reasonable performance in terms of data delivery ratio and control overhead. So in the following simulation experiments, we set both "Join Request" period in ODMRP and REQ_PERIOD in NSMP to two seconds. And recommended parameter contants were used for MAODV [12]. MAODV used link-layer unicast for all data transmissions while link-layer broadcast was used for NSMP and ODMRP data transmissions. And MAODV exploited link-layer ACK to maintain local connectivity.

Although IEEE 802.11 uses additional control packets for unicast, we did not count MAC layer control packets for control overhead for MAODV. We used the same movement scenarios and the same group scenarios.

4.2 Simulation Results

4.2.1 Experiments on NSMP Parameters

We first experimented on the impact of FLOOD_PERIOD on data delivery ratio and control overhead of NSMP. If FLOOD_PERIOD is long, less flooding route discovery operations will decrease control overhead, but data delivery ratio may drop due to more locally unrecoverable link failures. Since more sources with NSMP tend to increase mesh resilience, we used multicast groups of one source to isolate the effect of different FLOOD_PERIOD. We assumed five receivers per group and set maximum node speed to 20m/s. Different values of FLOOD_PERIOD (2, 20, and 100 seconds) were used with different transmission ranges. We also investigated the highest possible data delivery ratio as different transmission ranges led to different network topology.

In Figure 6, we can observe that packet delivery ratio improves up to 6% when FLOOD_PERIOD reduces from 100 seconds to 20 seconds. However, further reduction of FLOOD_PERIOD from 20 seconds to 2 seconds have marginal impact on packet delivery ratio. Figure 6 also illustrates that shorter transmission range causes lower data delivery ratio due to frequent network partition. Note that when FLOOD_PERIOD is set to 2 seconds, NSMP performs flooding route discovery only. When FLOOD_PERIOD is 20 seconds, locally unrecoverable link failures occur infrequently, so data delivery ratio is similar to that with 2 seconds. However, when FLOOD_PERIOD increases to 100 seconds, data delivery ratio drops significantly. And there was no significant difference in control overhead when FLOOD_PERIOD changes from 20 to 100 seconds. Considering data delivery ratio and control overhead, we set FLOOD_PERIOD to 20 seconds in the follwoing experiments.

We also simulated to study the impact of preferring forwarding nodes in establishing a reverse path. We used the following metric function:

$$Metric = (1 - a) \times FC + a \times NC, \quad 0 \leq a \leq 1.$$

And a path of smaller metric was selected as a reverse path. Here, a means relative weight for FC. A large a reduces the number of forwarding nodes and achieves route efficiency at the cost of a less resilient multicast mesh. In the other way, a smaller a tends to include more forwarding nodes but makes the mesh more robust. In the following simulation, we varied a from 0.4, 0.5 to 0.6. Transmission range was set to 250 meters, and the maximum node speed was set to 10m/s. A group had two sources, and we changed group sizes from 5 to 20.

Figure 7 shows the packet delivery ratio as a function of the group size varying $a = 0.4$, 0.5 and 0.6. In this figure, we can observe that the group size does not have great impact on the packet delivery ratio. We can also observe that the weight, a, has marginal effects on the packet delivery ratio. However, Figure 8, which shows the "Data packet/Total delivered packet" ratio as a function of the group size, reveals that larger a reduces the overhead of redundant packet transmissions significantly. When a increases from 0.4 to 0.6, the delivery ratio degradation is

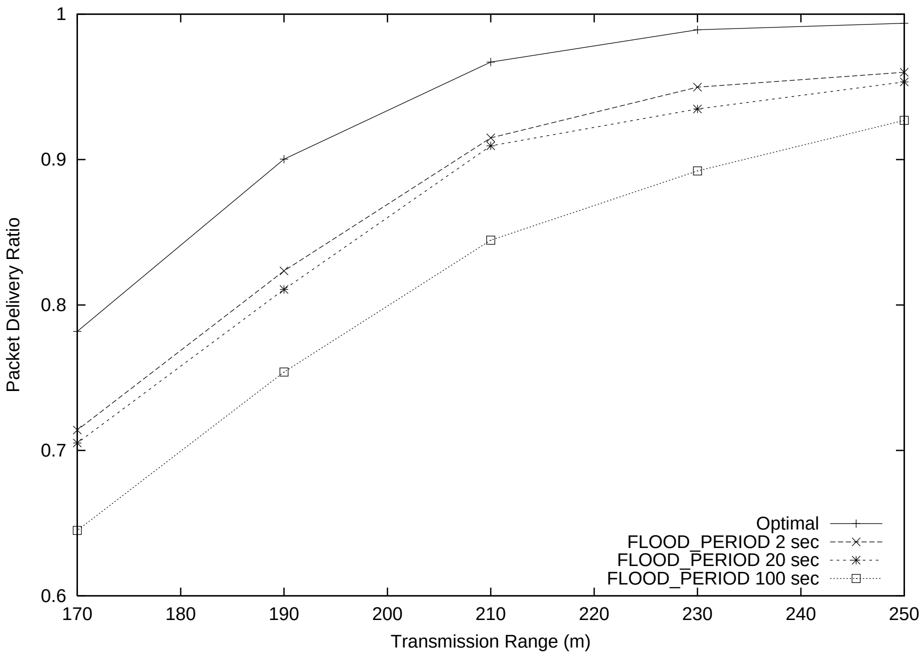

Figure 6: Data delivery ratio with different FLOOD_PERIOD

less than 1%. However, data transmissions are reduced by around 20%. We used 0.6 as the relative weight value in the following experiments.

In Figure 7, as group size increases, data delivery ratio of NSMP slightly drops. Also, data delivery ratio difference between different a's becomes smaller since more group members tend to lead to more resilient multicast meshes. In Figure 8, we can observe that data transmission per delivered data packet decreases as group members increase. It is because the number of forwarding nodes does not increase in proportion to the number of receivers since data path to each receiver can be shared.

4.2.2 Performance Comparison

We simulated and compared the performance of NSMP with those of ODMRP and MAODV. First, we investigated the impact of node mobility on each protocol. We varied the maximum node speed to study the resilence of protocols against the node mobility. A group has two source nodes and five receivers. Transmission range is set to 250 meters.

Figure 9 shows the packet delivery ratio as a function of the maximum node speed. All three protocols deliver data packets well when there is no node mobility. However, performance of tree-based MAODV degrades rapidly as the node speed increases. This performance degradation is due to frequent tree link failures. Since mesh structure of NSMP and ODMRP provides alternative paths, their data delivery ratios are higher in case of node mobility. We can observe that NSMP's data delivery ratio is comparable to that of ODMRP (within 1% difference). The performance gap between ODMRP and NSMP is due to the fact that NSMP performs local failure recovery while

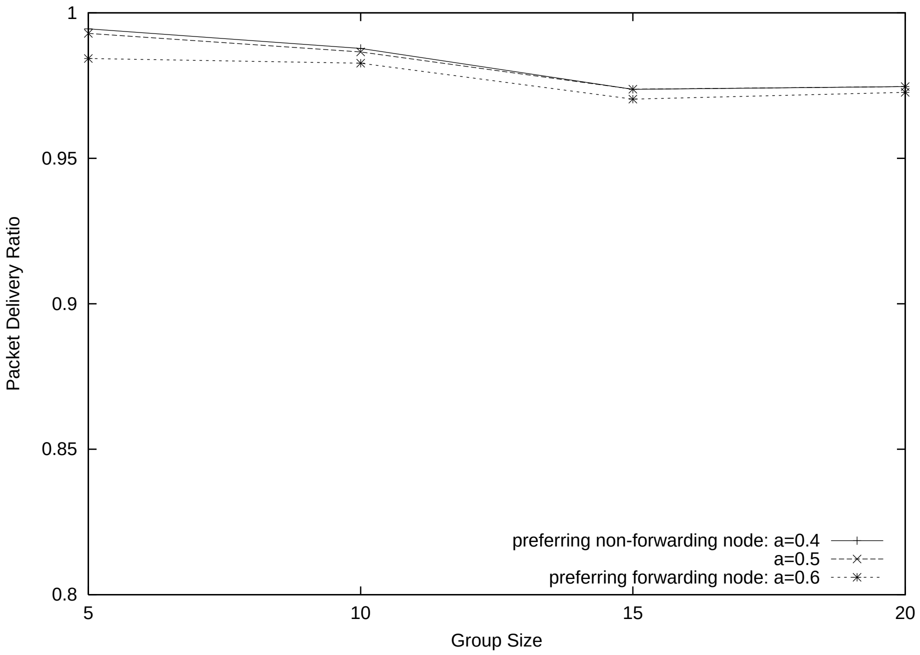

Figure 7: Data delivery ratio with different weight

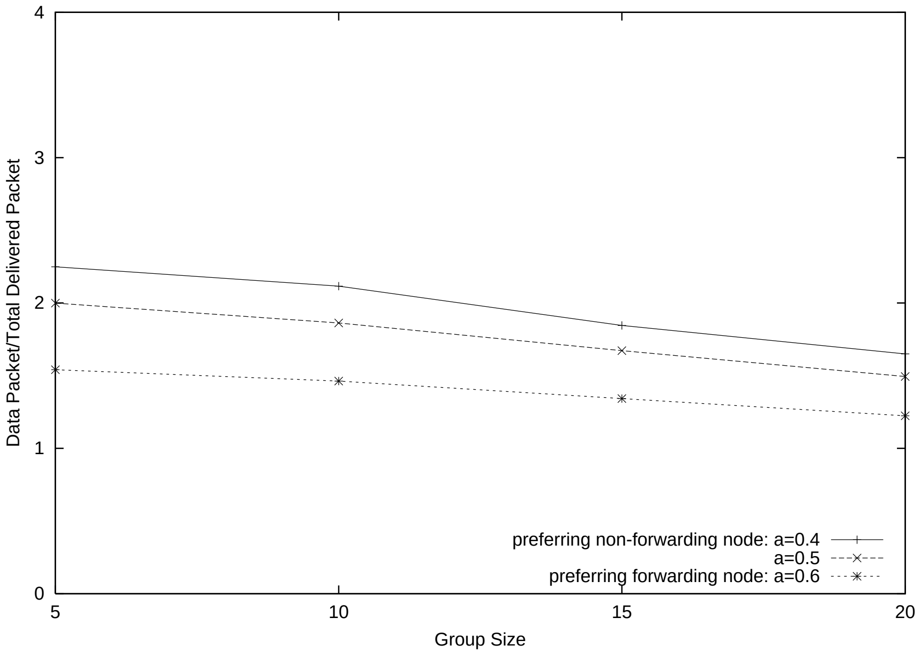

Figure 8: Data packet transmissions with different weight

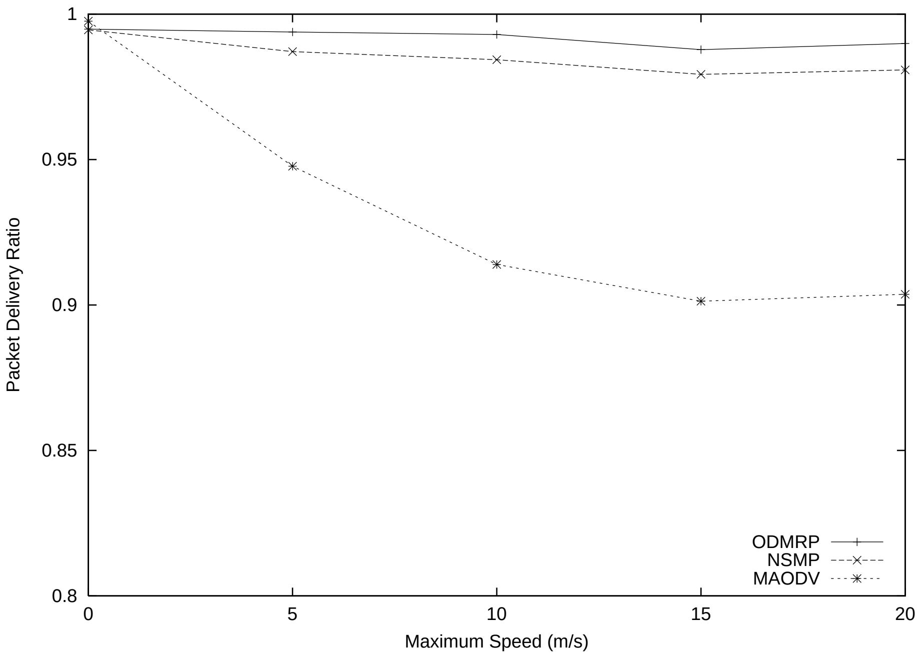

Figure 9: Comparison of data delivery ratio with mobility change

ODMRP performs global failure recovery.

Figure 10 illustrates data transmission overhead as a function of the maximum node speed. In general, MAODV has lower data transmission overhead than the other two protocols. However, the overhead of NSMP is as low as that of MAODV. This result endorses that the route created by NSMP is as efficient as that of tree-based protocols. The overhead of ODMRP is 20-30% larger than that of NSMP and MAODV. In Figure 10 and 11, we can conclude that NSMP achieves the robustness of mesh-based protocols while enjoying the efficiency of tree-based protocols. Control overhead is reported in Figure 11. MAODV incurs the lowest control overhead [1]. NSMP decreases control overhead by 5 - 15% compared to ODMRP using localized route discovery.

Figure 12 shows the packet delivery ratio as a function of the number of sources to investigate the scalability of NSMP. In the simulation, transmission range is fixed to 250 meters, maximum speed is set to 10 m/s, and a group has five receivers. ODMRP and NSMP perform better than MAODV in all cases. The performance of both NSMP and ODMRP increases as the number of source increases from one to two. We guess that the performance improvement is due to the increased redundancy of multicast meshes. However, as the number of source nodes increases further, the performance decreases due to increased traffic. However, because NSMP reduces both control and data packet transmissions, it exhibits less severe performance degradation in data delivery ratio than other two protocols. As overall network traffic grows higher, NSMP begins to show the best data delivery ratio.

[1] Note that we did not include control overhead for link-layer unicast.

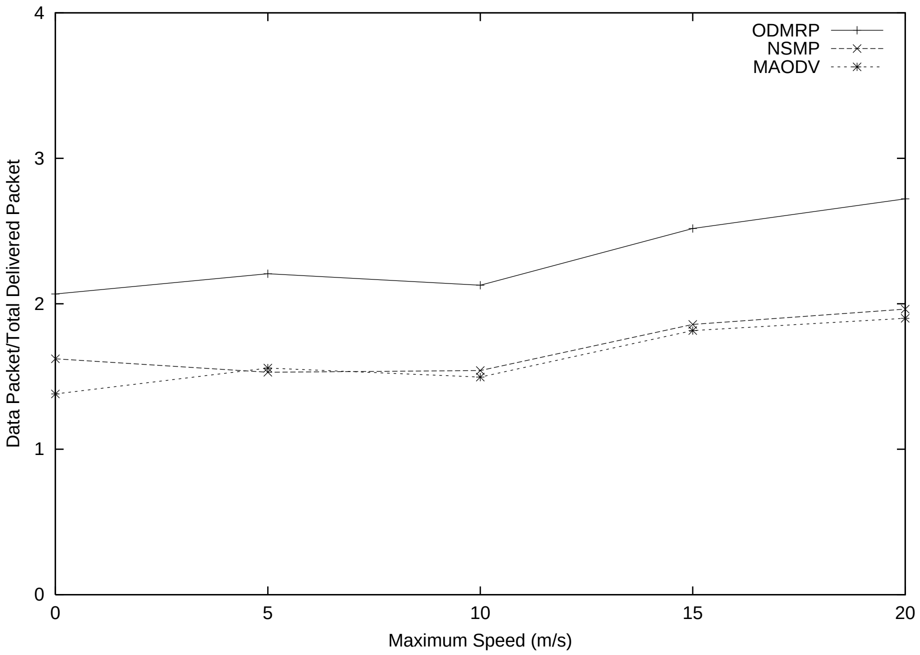

Figure 10: Comparison of data transmissions with mobility change

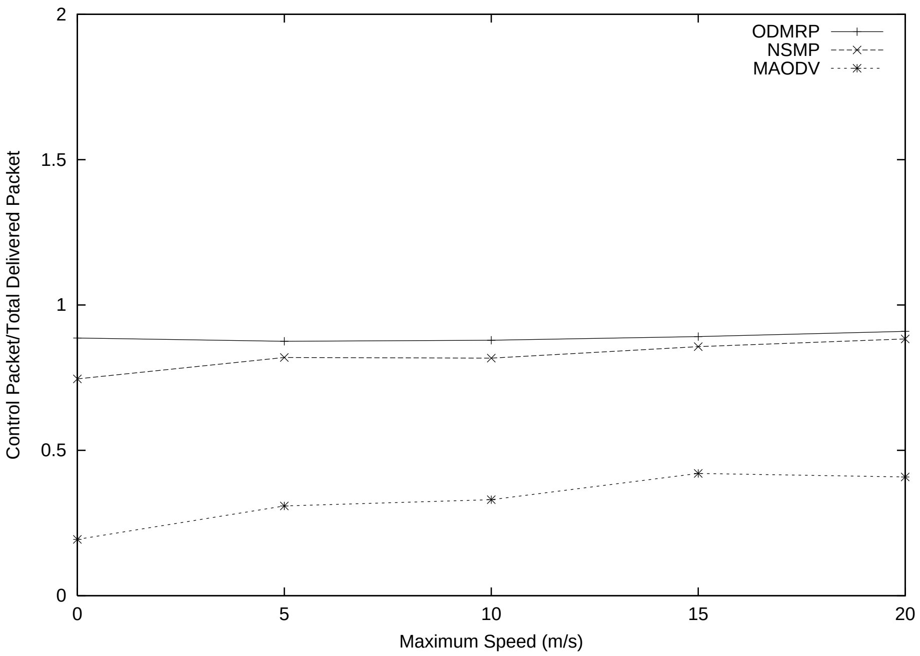

Figure 11: Comparison of control overhead with mobility change

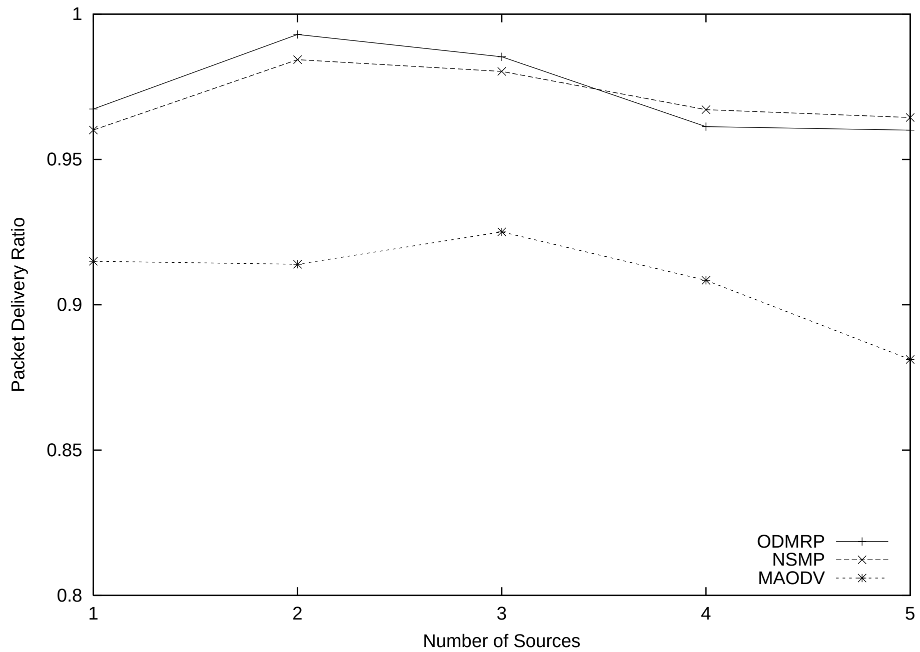

Figure 12: Comparison of data delivery ratio with increasing number of sources

5 Conclusions

This paper has proposed a new on-demand multicast routing protocol for ad hoc networks. The new routing scheme, NSMP, is based on multicast meshes and designed to minimize data transmissions and control overhead in maintaining the meshes. A key concept is to localize control messages to a small set of mesh nodes and group neighbor nodes and minimize the frequency of network-wide flooding. NSMP also attempts to improve route efficiency by giving preference to forwarding nodes in establishing a route. This leads to reduction in data packet transmissions and less contention in a network.

We simulated NSMP using *ns*-2 simulator, and simulation results reveal that NSMP effectively routes data packets. And NSMP substantially reduces control overhead and decreases data packet transmissions compared to ODMRP. Also, a mesh created by NSMP is efficient, and the number of data transmissions of NSMP is as low as tree-based MAODV. NSMP scales well with increasing group size and sources, and node mobility. Future research could be on how to adjust the period of route discovery packets under various mobility and traffic environments.

References

[1] D. Bertsekas, R. Gallager, "Data Network," pages 404-410, Second Ed. Prentice Hall, Inc., 1992

[2] C. Perkins, P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *ACM SIGCOMM*, Oct. 1994

[3] S. Murthy, J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *ACM Mobile Networks and Applications Journal, Special issue on Routing in Mobile Communication Networks*, 1996.

[4] V. Park, S. Corson, "Temporally-Ordered Routing Algorithm (TORA) Version 1," Internet draft, IETF, Aug. 1998.

[5] Josh Broch, David B. Johnson, David A. Maltz, "The Dynamic Source Routing in Ad Hoc Wireless Networks," In Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996

[6] Charles Perkins, Elizabeth M. Royer, Samir R. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing," Internet draft, IETF, Jun. 1999.

[7] C. K. Toh, "Long-lived Ad Hoc Routing based on the Concept of Associativity," Internet draft, IETF, Mar. 1999.

[8] G. Aggelou, R. Tafazolli, "RDMAR: A Bandwidth-efficient Routing Protocol for Mobile Ad Hoc Networks," *Proceedings of The Second ACM International Workshop on Wireless Mobile Multimedia (WoWMoM)*, Seattle, Washington, Aug. 1999.

[9] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, ACM, Dallas, TX, Oct. 1998.

[10] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, M. Degermark, "Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks," *MobiCom '99*, Aug. 1999.

[11] S. Corson, J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," RFC 2501, IETF, Jan. 1999.

[12] Elizabeth Royer, Charles E. Perkins "Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol," *MobiCom '99*, Aug. 1999.

[13] S. Lee, W. Su, M. Gerla, "Ad hoc Wireless Multicast with Mobility Prediction," *IEEE ICCCN '99*, Boston, MA, Oct. 1999.

[14] J. J. Garcia-Luna-Aceves, E. L. Madruga, "The Core-Assisted Mesh Protocol," *IEEE Journal on Selected Area in Communications, Special Issue on Ad-Hoc Networks*, Vol. 17, No. 8, Aug. 1999.

[15] C. W. Wu, Y. C. Tay, C-K. Toh, "Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) Functional Specification," Internet draft, IETF, Nov. 1998.

[16] E. Bommaiah, M. Lui, A. McAuley, R. Talpade, "AMRoute: Adhoc Multicast Routing Protocol," Internet draft, IETF, Aug. 1998.

[17] S. Lee, W. Su, J. Hsu, M. Gerla, R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols," *INFOCOM 2000*, Mar. 2000.

[18] R. Castaneda, S. Das, "Query Localization Techniques for On-Demand Routing Protocols in Ad Hoc Networks," *MobiCom '99*, Aug. 1999.

[19] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, Jang-Ping Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *MobiCom '99*, Aug. 1999.

[20] Kevin Fall, Kannan Varadhan, editors, "ns notes and documentation," The VINT Project, UC Berkeley, LBL, USC/ISI and Xerox PARC, Nov. 1997. Available at http://www-mash.cs.berkeley.edu/ns/.

[21] "The CMU Monarch Project's Wireless and Mobility Extensions to ns," The CMU Monarch Project, Aug. 1999. Available at http://www.monarch.cs.cmu.edu/.