Neighbor Supporting 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

Abstract— **An ad hoc network is a multi-hop wireless network formed by a collection of mobile nodes without the intervention of fixed infrastructure. Limited bandwidth and a high degree of mobility require that routing protocols for ad hoc networks 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 failure recovery and mesh 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.**

I. 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 of it 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.

This work was supported in part by the Brain Korea 21 Project and grant No. (1999-1-302-005-3) from the interdisciplinary Research program of the KOSEF.

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] are generally more efficient in terms of data transmission than mesh-based protocols, but they are not robust against topology changes because there is no alternative path between a source and a destination. Mesh-based multicast protocols [13], [14] 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].

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. Broadcasts are expensive operations in ad hoc networks [18]. 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 effectively delivers around 97% of data packets and is robust against frequent toplogy changes. Moreover, data packet transmissions and control message exchanges are reduced by 15-20% compared to existing ad hoc multicast routing protocols. We also observed that NSMP reduces the average packet delay by 10%.

The rest of this paper is organized as follows. Section II contains an overview of NSMP, and a more detailed description of NSMP is presented in section III. Section IV provides results of simulation experiments, and section V concludes the paper.

II. A New Multicast Routing Protocol

A. 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 mesh infrastructure. It is known that the mesh structure is more robust against topology changes than the tree structure [14], [17]. However, the tree structure is better than the mesh structure in terms of packet transmissions. 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 and further to lower end-to-end delay.

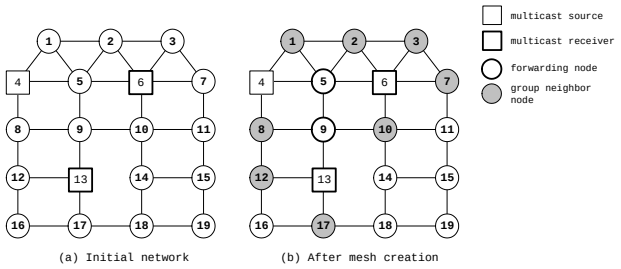

Fig. 1. Multicast mesh creation

B. 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*.

Fig. 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 and node 9 becomes a forwarding node. Fig. 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 Fig. 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 predefined timeout period. Section III shows detailed procedures of how a multicast mesh is built and a node becomes a group neighbor.

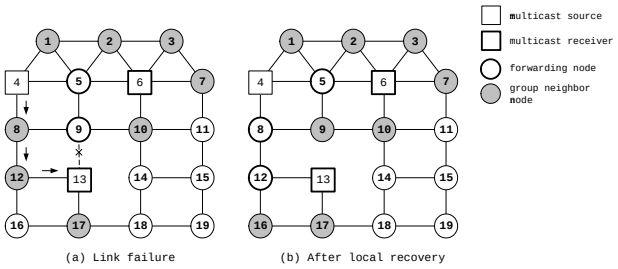

Fig. 2. Multicast mesh maintenance

Type	Sequence Number	Group Address	Source Address	Upstream	FC	NC

Fig. 3. Packet header of NSMP

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

Fig. 4. Routing table used in NSMP

C. Multicast Mesh Maintenance

C.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 II-B. 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 Fig. 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 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 Fig. 2 (b). Note that more than 30% of the nodes (i.e. six nodes) in Fig. 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 Fig. 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 IV show that local route discovery is effective under various environments.

C.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.

D. 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 Fig. 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.

III. Detailed Description of NSMP

A. Data Structures and Packet Header

Fig. 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 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*. Fig. 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 Fig. 5. Every source node needs to maintain a *SourceList* that consists of source addresses of the same group.

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

(a) DataCache

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

(b) ReqCache

Fig. 5. Caches used in NSMP

TABLE I

SUMMARY OF NODE BEHAVIORS WHEN A ROUTE DISCOVERY PACKET ARRIVES

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

B. 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 II-C, 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 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 III-C.

C. 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, $(1-a)\times FC+a\times NC$, where $0 \leq a \leq 1$ is the relative weight. 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.

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.

D. 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 I summarizes node behaviors when it receives route discovery packets.

E. 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.

F. 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 Fig. 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 *Req-Cache*. MEM_REQ uses a *ttl* field. All nodes that receive a MEM_REQ packet relay the packet only if *ttl* value 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 SRC_REQ packets. 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 Fig. 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.

G. 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.

IV. Performance Simulation

A. 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 [19] and recently extended to provide simulation support for ad hoc networks by the MONARCH project [20] at Carnegie Mellon University. Reference [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 in a simulation move according to the "random waypoint" model as in [9] without pause time. A multicast source generates 512-byte data packets with constant bit rate (CBR) of two 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 determines movements of 50 mobile nodes, and the speeds of mobile nodes are uniformly distributed up to a maximum speed. Group scenario files determine which nodes are receivers or sources and when they join and leave a group. A receiver randomly joins a group between 0 and 60 seconds and leaves the group between 240 and 300 seconds. Multicast sources start and stop sending packets in the same fashion. Each group scenario file has one multicast group. We used the average of 15 experiments (combination of three group scenarios and five movement scenarios).

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 and Control Packets per Data Packet Delivered**: This metric represents the routing efficiency and the degree of control overhead, respectively.
- **Average End-to-End Delay**: The average time between a transmission of a data packet and a successful reception at a receiver

We compare the performance of NSMP with that of ODMRP. It was assumed that no nodes were equipped with GPS, so a source in ODMRP periodically flooded "Join Request" packets. Both "Join Request" period in ODMRP and REQ_PERIOD in NSMP were set to two seconds for the fair comparison. We also used the same movement scenarios and the same group scenarios.

B. Simulation Results

We first experimented on the impact of FLOOD_PERIOD on data delivery ratio and control overhead of NSMP. We set maximum node speed to 40m/s and assumed that there

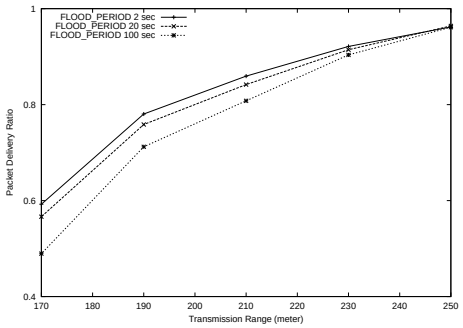

Fig. 6. Data delivery ratio with different FLOOD_PERIOD

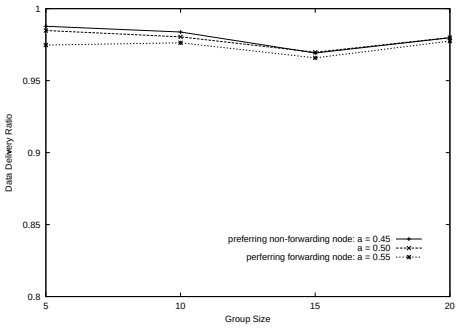

Fig. 7. Data delivery ratio with different weight

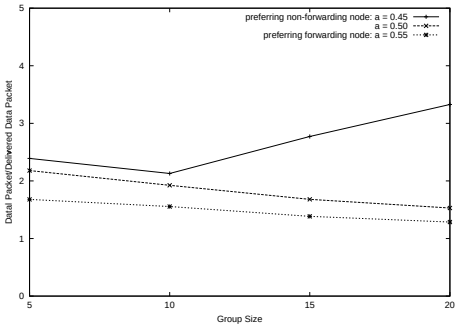

Fig. 8. Data packet transmissions with different weight

were two group sources and five receivers. Different values of FLOOD_PERIOD (2, 20, and 100 seconds) were used with different transmission ranges. In Fig. 6, we can observe that data delivery ratio does not depend much on the frequency of flooding route discovery when network connectivity is high. It is because network partitions and locally unrecoverable link failures rarely occur in such environments. FLOOD_PERIOD has a greater influence on the data delivery ratio at lower network connectivity. When FLOOD_PERIOD is 20 seconds, data delivery ratio slightly drops, compared to when FLOOD_PERIOD is 2 seconds. But the difference is around 1%. However, when FLOOD_PERIOD increases from 20 seconds to 100 seconds, data delivery ratio drops significantly. Because the control packet overhead decreases significantly when FLOOD_PERIOD changes from 2 to 20 seconds, and the packet delivery ratios at FLOOD_PERIOD 2 and 20 seconds are almost the same, we set FLOOD_PERIOD to 20 seconds in the following experiments.

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

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

And a path of smaller metric was selected as a reverse path. Thus, $a > 0.5$ improves route efficiency while $a < 0.5$ results in a more robust mesh. Because a values smaller or greater than 0.5 give almost the same results respectively, we simulated with values of 0.45, 0.5 and 0.55. Transmission range was set to 250 meters, and the maximum speed was 10m/s. A group had two sources, and we changed group sizes from 5 to 20. Fig. 7 and 8 show that when $a = 0.45$, data delivery ratio slightly increases while more forwarding nodes lead to more data transmissions. Especially, when there are 20 receivers, there are two times as many data transmissions as when $a = 0.50$. When $a = 0.55$, there is less than 1% data delivery ratio degradation, but data transmissions are reduced by around 20%. We can also observe in this experiment that NSMP scales well with increasing group size. We used 0.55 as the relative weight value in the following experiments.

We also compared the performance of NSMP with that of ODMRP. In the first set of simulations, we changed the number of source nodes to investigate the protocol scalability. Transmission range is 250 meters, maximum speed is 10m/s, and a group has five receivers. In Fig. 9, NSMP is as effective as ODMRP in data delivery ratio, and the difference is less than 1%. As shown in Fig. 10, however, control overhead decreases by more than 15% regardless of the number of sources. Route efficiency of NSMP also enables large improvement (up to 35%) in terms of data transmissions, as shown in Fig. 11. Fig. 12 shows that the decreased number of packet transmissions leads to reduced end-to-end delay. The differences of data transmissions and end-to-end delay between NSMP and ODMRP increase as the number of sources increases. We can also observe that NSMP scales well with increasing number of sources.

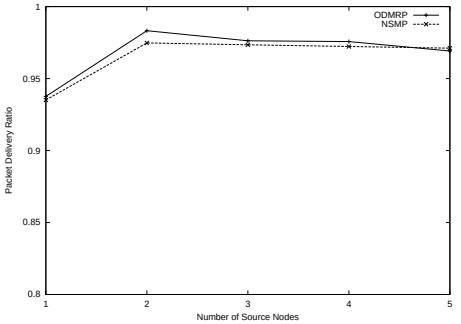

Fig. 9. Comparison of data delivery ratio (source number change)

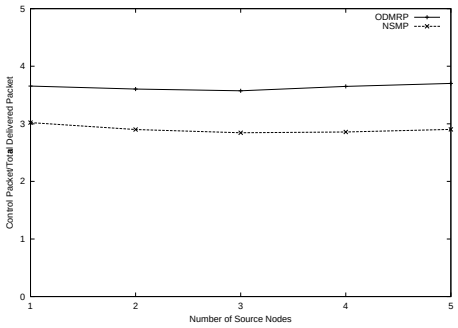

Fig. 10. Comparison of control packet transmissions (source number change)

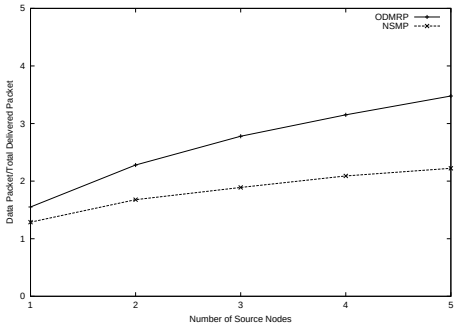

Fig. 11. Comparison of data packet transmissions (source number change)

In order to investigate the impact of mobility on NSMP, we changed the maximum speed of mobile nodes. Fig. 13 and 14 show the result. In this experiment, a group has three sources and five receivers. In Fig. 13, delivery ratio difference between NSMP and ODMRP is less than 1%. However, Fig. 14 shows that data transmissions in NSMP are reduced by more than 30%, compared to those in ODMRP. In Fig. 14, the amount of data transmissions slightly increases as the nodes move faster. It is because more frequent link failures cause more unnecessary nodes to be forwarding nodes temporarily. From this experiment, we can conclude that NSMP does not show noticeable performance degradation due to high mobility.

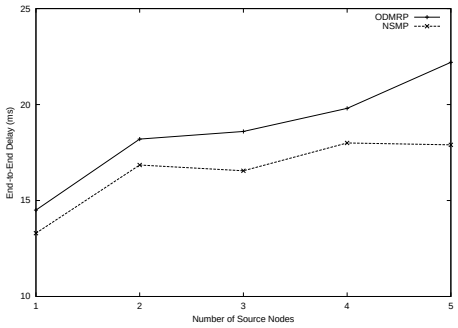

Fig. 12. Comparison of end-to-end delay (source number change)

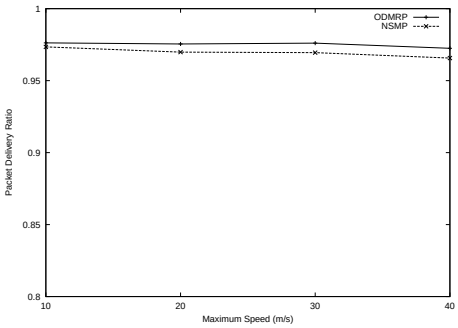

Fig. 13. Comparison of data delivery ratio (speed change)

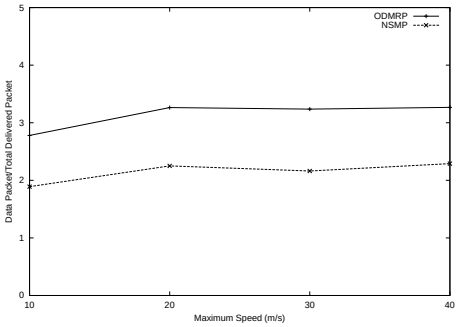

Fig. 14. Comparison of data packet transmissions (speed change)

V. 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 further to decreased average delay due to less contention in a network.

We simulated NSMP using *ns*-2 simulator, and simulation results reveal that NSMP effectively routes data packets. NSMP substantially reduces control overhead and decreases data packet transmissions compared to ODMRP. Also, NSMP scales well with increasing group size and sources and does not show performance degradation in case of high mobility. Future research could be on how to adjust the period of route discovery packets.

References

[1] D. Bertsekas, and 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, and 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, and S. Corson, “Temporally-ordered routing algorithm (TORA) Version 1,” Internet draft, IETF, Aug. 1998.
[5] Josh Broch, David B. Johnson, and 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, and 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, and 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, and J. Jetcheva, “A performance comparison of multi-hop wireless ad hoc network routing protocols,” *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, ACM, Dallas, TX, Oct. 1998.
[10] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, “Scenario-based performance analysis of routing protocols for mobile ad-hoc networks,” *MobiCom ’99*, Aug. 1999.
[11] S. Corson, and J. Macker, “Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations,” RFC 2501, IETF, Jan. 1999.
[12] Elizabeth Royer, and Charles E. Perkins “Multicast operation of the ad-hoc on-demand distance vector routing protocol,” *MobiCom ’99*, Aug. 1999.
[13] S. Lee, W. Su, and M. Gerla, “Ad hoc wireless multicast with mobility prediction,” *IEEE ICCCN ’99*, Boston, MA, Oct. 1999.
[14] J. J. Garcia-Luna-Aceves, and 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, and 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, and R. Talpade, “AMRoute: adhoc multicast routing protocol,” Internet draft, IETF, Aug. 1998.
[17] S. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, “A performance comparison study of ad hoc wireless multicast protocols,” *INFOCOM 2000*, Mar. 2000.
[18] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu, “The broadcast storm problem in a mobile ad hoc network,” *MobiCom ’99*, Aug. 1999.
[19] Kevin Fall, and 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/.
[20] “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/.