CQMP: A Mesh-based Multicast Routing Protocol with Consolidated Query Packets

Harleen Dhillon and Hung Q. Ngo
Computer Science and Engineering,
State University of New York at Buffalo,
201 Bell Hall, Amherst, NY 14260, USA.
{dhillon, hungngo}@cse.buffalo.edu

***Abstract*— We propose a mesh-based multicast routing protocol for wireless ad hoc networks. The protocol retains all of the advantages of the on-demand multicast routing protocol (ODMRP) such as high packet delivery ratio under high mobility, high throughput. Moreover, the protocol significantly reduces control overhead, one of the main weaknesses of ODMRP, under the presence of multiple sources. This feature is a crucial contributing factor to the scalability of multicast routing for mobile ad hoc networks. The results are experimentally verified. It is shown that in the presence of high number of sources, our protocol reduces the control packet load by upto 30 percent, increases the multicast efficiency by 10 to 20 percent and improves the data delivery ratio of ODMRP.**

I. INTRODUCTION

Mobile ad hoc networking has emerged as one of the major sub-areas of communication networks, since the potential set of ad hoc network applications is enormous, ranging from mobile conferencing, emergency services, to battle field communications. Multicasting is probably the most important communication primitive for these applications. One of the major challenges faced by researchers in this area is to design an effective and scalable multicast routing protocol under the mobile and infrastructureless conditions of ad hoc wireless networks. As network nodes' mobility increases, maintaining persistent network states to assure high packet delivery ratio and high throughput becomes more difficult. It is intuitively obvious that control overhead is a price we have to pay for mobility. However, control overhead is also one of the major factors affecting power consumption, a very important resource for ad hoc network nodes. In some applications, in fact, energy is entirely non-renewable ([1], [2]). Consequently, devising a multicast routing protocol with low control overhead, yet highly effective under dynamic network conditions, is a very important problem in mobile ad hoc networking.

There are two main design philosophies for routing protocols under mobile ad hoc networks: *proactive* (distance vector or link-state type), and *reactive* (also referred to as *on-demand*).

Proactive protocols require all nodes to periodically exchange information to maintain global routes. Distance vector and link-state routing protocols have been well-studied for infrastructured networks. Hence, the advantage is that we can inherit a lot from existing implementations and literature. Unfortunately, proactive protocols often suffer from excessive control overhead due to frequent updates when network mobility is high. Both distance vector and link-state based routing protocols also impose linear growth in routing table sizes. Given that ad hoc network nodes have scarce resources, large routing tables are not desirable, affecting the scalability of the protocol.

Reactive (on-demand) protocols have been widely seen as the solution to the control overhead problem. The basic idea is to discover routes only when needed. Small QUERY/REPLY packets are used for this purpose. For unicasting, many of such protocols have been proposed in recent years: ad hoc on-demand distance vector (AODV, [3]), dynamic source routing (DSR, [4]), temporally ordered routing algorithm (TORA, [5]), associativity based routing (ABR, [6]), and signal stability based adaptive routing (SSA, [7]). There is also an attempt to combine reactive and proactive approaches in the zone routing protocol (ZRP, [8], [9]).

In general, on-demand protocols impose lower overhead and storage even in large networks. However, as mobility increases, control overhead gets higher and higher due to periodic flooding and route breakdowns. Some protocols even generate more overhead than actual throughput [10]. Thus, it is still a very challenging problem to devise scalable routing protocols for dynamic networks.

Many multicast routing protocols have been proposed for ad hoc networks in recent years (see [11], for example, for a survey). We shall review some of these in the next section. Two major classes of multicast routing protocols are *tree-based* and *mesh-based*.

Tree-based schemes establish a single path between any two nodes in the multicast group. These schemes require a minimum number of copies per packet to be sent along the branches of the tree. Hence, they are bandwidth efficient. If there is only one source, then oftentimes there is only a minimal number of nodes involved in the routing. Hence, tree-based schemes could also be relatively power-efficient. However, as mobility increases, link failures trigger the reconfiguration of the entire tree. When there are many sources, one either has to maintain a shared tree, losing path optimality, or maintain multiple trees resulting in storage and control overhead. Examples of tree-based schemes include: ad hoc multicast routing protocol (AMRoute, [12]), ad hoc multicast routing utilizing increasing ID-numbers protocol (AMRIS, [13]), bandwidth efficient multicast protocol [14], multicast ad hoc on-demand distance vector routing protocol (MAODV, [15]), and multicast core-extraction distributed ad hoc routing protocol (MCEDAR, [16]).

Mesh-based schemes establish a mesh interconnecting the source and the destinations. They are more resilient to link failures, and thus, to mobility. In multiple source cases, they can provide better paths than a shared tree. The major disadvantage is that mesh-based schemes introduce higher redundancy since multiple copies of the same packet are disseminated through the mesh, resulting in bandwidth wastage and power-inefficiency. This is the price one has to pay for higher packet delivery and throughput under highly mobile conditions. Examples of mesh-based schemes include: on demand multicast routing protocol (ODMRP, [17]), forwarding group multicast protocol (FGMP, [18]), core assisted mesh protocol (CAMP, [19]), neighbor supporting ad hoc multicast routing protocol (NSMP, [20]), location-based multicast protocol [21], and dynamic core-based multicast protocol (DCMP [22]).

Of these protocols, ODMRP is probably one of the most well-studied protocols. The major advantage of ODMRP is that it produces high packet delivery ratio and throughput even under highly mobile network conditions. The disadvantage is that the control overhead introduced also grows higher and higher. Several earlier attempts have been made to reduce ODMRP's control overhead ([22], [23]).

In this paper, we propose a mesh-based on-demand multicast routing protocol with an idea of "query packet consolidation" to address this scalability problem. We shall refer to our protocol as **CQMP** (for Consolidated Query-packet Multicast Protocol) henceforth. Our ideas are then validated by extensive simulations, which show that as the network scales, the control overhead can be much reduced than that experienced by ODMRP in similar situations, while maintaining comparable levels of throughput.

The rest of the paper is organized as follows. Section II overviews several related works in more details. Section III discusses the intuitions behind our protocol. Section IV presents the new protocol in its full technical details. Section V gives the simulation results validating our protocol. Lastly, Section VI concludes the paper.

II. Related Work

Mesh-based protocols transmit data to receivers by creating a forwarding mesh. A mesh between sources and receivers enables redundancy in the routes, which increases the robustness of the paths. In ODMRP [17], a source periodically floods an advertising packet in the network. A receiver responds to this packet by using backward learning. The nodes on the path from the receiver to the source form a mesh of forwarding nodes for the multicast group. FGMP [18] is similar to ODMRP. The major difference lies in who initiates the flooding. Whereas in ODMRP, the flooding of group membership advertising packets in the network in initiated by a source, in FGMP, the receivers are the entities that perform this flooding. In addition, in FGMP, a source has to keep track of all receivers in a multicast group. Both of these protocols result in considerable control overhead due to frequent flooding.

CAMP [19] attempts to eliminate this flooding by the use of core nodes. Instead of each source sending advertising packets to the network, in CAMP, each core disseminates to the network the mappings of multicast addresses to one or more core addresses. CAMP, however, assumes the availability of routing information from a unicast routing protocol. This unicast routing protocol is also required to provide correct distances to known destinations within a finite amount of time. CAMP also assumes the existence of a beaconing protocol, which may be embedded into the unicast routing protocol. In addition, CAMP relies on the associated routing protocol to work correctly in the presence of router failures and network partitions.

NSMP [20] is another mesh-based protocol that tries to reduce flooding. Like ODMRP, NSMP operates independently of the unicast routing protocol. It reduces the routing overhead by localizing route discovery and maintenance operations. For an initial route establishment or a network partition repair, NSMP performs flooding route discovery in which control messages are broadcast by all nodes. Since routine path maintenance usually occurs much more frequently than the initial path establishment, the saving by localized path maintenance could be sizable.

In Location-Based Multicast protocol [21], location information is used to limit the flooding in the network. This thus necessitates the use of a global positioning system or similar tools. Based on location of the multicast region, *forwarding zones* are defined. Only nodes in the forwarding zone forward a multicast packet.

DCMP [22] attempts to reduce the flooding in the network by specifying some sources as passive. Broadcast Medium Window [24] uses downstream congestion information to determine whether or not to send or transmit an advertising packet. ODMRP-MPR [23] uses two-hop neighborhood information to select a set of multicast relay nodes [25] that perform network flooding of advertising packets.

III. Intuitive introduction to our modifications

ODMRP provides a high packet delivery ratio even at high mobility, but at the expense of heavy control overhead. It does not scale well as the number of senders and traffic load increases [26]. Since every source periodically floods advertising packets called Join Queries through the network, congestion is likely to occur when the number of sources is high. We present an algorithm that consolidates Join Queries together in an attempt to reduce the total number of Query transmissions. Where there exist sources in a network that relay Join Queries within a close interval of each other, there is a certain amount of redundancy that can be reduced. This motivates our ideas.

IV. DETAILED TECHNICAL DESCRIPTIONS

Each source is identified by a unique identifier called the NODEID. Each source also generates a unique SEQUENCE NUMBER that is appended into all packets that it originates. A node receiving a packet from a source temporarily saves the pair (SENDER ID, SEQUENCE NUMBER) from the packet into a message cache to avoid duplicate packet processing.

An active source periodically transmits an advertising packet called QUERY to the network. We will use the terms QUERY and JOIN QUERY interchangeably in the rest of this document. The QUERY is flooded at regular intervals and for two reasons: (a) to update existing routes to match the current network conditions, and (b) to enable new receivers to create a route to itself. Before transmitting the QUERY packet, the source generates two sequence numbers. We will call them CURRENTSEQ and NEXTSEQ. Fig. 1 shows the format of a QUERY packet. The QUERY SEQUENCE NUMBER field is set to be the CURRENTSEQ. The SENDER ID field is filled by the NODEID of the source. These two fields will not change throughout the lifetime of this QUERY. The NUMSOURCES field refers to the number of sources whose information is contained in this QUERY. At this point, it would be just 1. The LAST HOP field is its own ID. In addition to these fields, the first SOURCE ROW of the QUERY packet is filled as follows. The SOURCE ID is set to be its own ID and HOPCOUNT, which refers to the number of hops traversed by the QUERY, is set as 1. The NEXTSEQ is used to relay information to the nodes receiving this QUERY about the next query transmission. INT refers to the interval after which the source will send another QUERY. Hence, a node processing this information can expect to receive another QUERY from this source with sequence number NEXTSEQ and after INT time units. The number NEXTSEQ is saved by the source for later use in consolidation.

NumSources	Sender Id	Query Sequence Number	Last Hop Id			
Source Row #	Source Id	Multicast Group Id	CurrentSeq	NextSeq	HopCount	INT
...	...	...	...	...	...	...

Fig. 1. Format of a QUERY packet

After these fields are filled, the QUERY is transmitted to the network. When a node receives this QUERY, it determines whether it can consolidate into this QUERY information about other sources from which it is expecting to hear a QUERY. This process is described in the following sub-section.

A. *Query consolidation by intermediate nodes*

When a node receives a QUERY packet, it first compares the pair (SENDER ID, SEQUENCE NUMBER) from the QUERY with the entries in its message cache. If there is a match, the QUERY is discarded as a duplicate. If not, it is processed as follows.

From each Source Row in the QUERY, the node retrieves the pair (SOURCE ID, CURRENTSEQ) and checks it against its message cache. If there is a match in the cache, the node goes on to the next row. Otherwise, the node saves the Source ID, NextSeq and INT values from the Source Row, and the Last Hop from the QUERY as an entry into a table. This table is used as the **Routing Table (RT)** for the REPLIES sent back toward the source. In addition, the (SOURCE ID, CURRENTSEQ) pair is saved in its message cache.

The node now goes through its RT (Routing Table) to determine if it is expecting any QUERIES from the same Last Hop as carried by the received QUERY within a small TIME-INTERVAL. It would know this if, for some entry in its RT, all of the following conditions apply:

- The NEXTSEQ field is not null.
- INT would expire within this TIME-INTERVAL.
- LAST HOP in the received QUERY packet is the same as the LAST HOP in the RT entry.

If all these conditions apply, the node will add another Source Row in the QUERY packet and increment its NumSources. While the Source ID for the new row is taken from the same field from the corresponding entry in the RT, the value of its CurrentSeq is set as equal to the NextSeq from the RT entry. The NextSeq field in the new Source Row is left empty. Information from any applicable (as determined by the conditions described above) entries in the RT is thus added into the Source Rows, and the NumSources is incremented every time. If information about a source entry in the RT is appended into the QUERY, the NextSeq field for that source entry in the RT is made null. This step is performed to ensure that information about the same source and sequence number pair is not appended into another QUERY. Now the node sets the Last Hop ID field in the QUERY to itself and transmits the QUERY.

This process continues until the QUERY reaches a multicast receiver node. At this point, the QUERY may contain more than one Source Row. The receiver goes through each and every Source Row entry in the QUERY, and builds and transmits a REPLY packet based upon matched entries. When any node receives a REPLY packet, it checks if the next node Id in any of the entries in the REPLY matches its own. If so, it realizes that it is on the way to a source, and sets a flag indicating that it is part of the FORWARDING GROUP for that multicast group. It then builds and broadcasts its own REPLY packet. When a REPLY reaches a source, a route is established from the source to the receiver. The source can now transmit data packets towards the receiver. A Forwarding Group node will forward any data packets received from a member for that group.

The example in Fig. 2 demonstrates the process. The connected lines between two nodes indicate that they are within transmission range of each other. The nodes 1 and 3 are sources. Let's assume that both sources start flooding their QUERIES at the same time with sequence number 100 incremented by one with every new QUERY. At this time, there would only be one Source Row in the packet that would consist of the sender's own ID as Source ID, 100 as CurrentSeq and 101 as NextSeq.

The QUERY packet from source 1 reaches node 2 first. Node 2 first checks the pair (1,100), i.e. the (SENDER ID, SEQUENCE NUMBER) from the QUERY against its message cache. Since there is no match found, node 2 processes the packet. From the first Source Row of the Query, node 2 saves in its RT the INT value which will tell it when to expect the next QUERY from source 1, the NextSeq 101 and the Last Hop ID. In this case, the Last Hop ID toward source 1 would be 1. It also saves the (SOURCE ID, CURRENTSEQ) pair - which is (1, 100) - in its message cache and transmits the QUERY to its neighbors with the Last Hop ID set as its own NodeID. This process is repeated by all nodes receiving this Query. As shown in the figure, node 7 hears the QUERY from both sources via node 6 and nodes 8 and 9 hear the Queries via node 7.

After the INT is over, both sources flood QUERY packets with sequence numbers 101 and NextSeq 102. Let's assume that node 7 first hears the QUERY from source 1 via node 6. Node 7 processes the packet as described. It now checks if it is expecting any other Queries via node 6 within TIME-INTERVAL. From its routing table, it realizes that it should be receiving a QUERY from source 3 with NextSeq 101 through node 6. It thus appends a second row in the QUERY from source 1. In this row, it sets the Source Id as 3, CurrentSeq as 101, and leaves the rest of the fields blank. It then increments the NumSources, sets its own NodeID as the Last Hop and transmits the QUERY to its neighbors. Node 7 also saves the pair (3, 101) in its message cache.

The QUERY that nodes 8 and 9 receive from node 7 will have two source rows, one representing each source. Nodes 8 and 9 will thus process both rows and save the pairs (1,101) and (3,101) in their respective message caches. Now, when node 7 receives the QUERY from source 3 with sequence number 101, the pair (3,101) would already be present in 7's message cache. Node 7 thus determines this QUERY to be duplicate and drops it. There is no loss in this dropping since nodes 8 and 9 have already processed the pair (3,101). Thus, there would be three less QUERY transmissions here than if there were no consolidation.

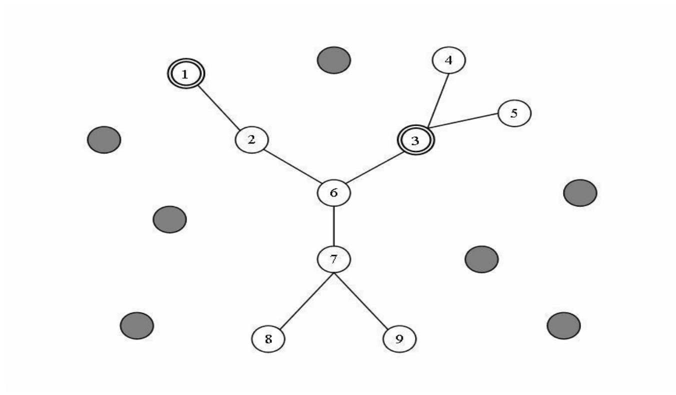

Fig. 2. Query consolidation at any node

B. Query consolidation by source

When a source receives a QUERY from another source, it processes the packet just as described above for any node. The following is performed in addition.

Before transmitting the QUERY, the source checks its INT to determine if it would expire within TIME-INTERVAL. What that means is the source checks if it is about to create and transmit its own QUERY between now and TIME-INTERVAL. If so, it adds one more row to the QUERY. This row consists of its own NodeID as Source Id, and the CurrentSeq field is filled by the NextSeq that it generated and saved when it sent its last QUERY. The rest of the fields are left empty. The NumSources field in the QUERY is incremented and the QUERY is then transmitted with the LAST HOP set as its own NodeID.

If within this short interval prior to creating and transmitting its own QUERY, more than one QUERY from different sources is received, the source would add its information into each of them. When the INT is over, the source sets its CurrentSeq equal to its NextSeq and generates a new NextSeq. The source then creates and transmits a new QUERY as described previously.

When this QUERY is received by a node, it may already have consolidated the information represented by this QUERY with another QUERY or have received the same information from a Source Row of another QUERY. If any of these is true, then the node would have the (SENDER ID, SEQUENCE NUMBER) pair carried by the QUERY in its cache and the QUERY would be treated as a duplicate. If the node has no information in its cache about this pair, then the QUERY is processed as described previously in this section.

The example in Fig. 3 demonstrates this kind of QUERY consolidation. Nodes 1 and 7 are the sources whose the sequence numbers start from 100 and are incremented by 1 with every new QUERY.

Source 1 sends a QUERY with (1,100) as the (SENDER ID, SEQUENCE NUMBER) pair. The NextSeq is 101. Let's assume that the QUERY from source 1 reaches source 7 within TIME-INTERVAL before source 7 has sent its first Query. Source 7 then appends another Source Row in the QUERY from source 1. This Source Row has Source Id as 7, and CurrentSeq as source 7's NextSeq, which should be 100. The rest of the fields in the Source Row are left empty. Source 7 also increments the NumSources field. It now sets itself as the Last Hop and transmits the QUERY to its neighbors.

Source 1's QUERY that nodes 8, 9 and 10 receive from node 7 will have two source rows, one representing each source. Nodes 8, 9 and 10 will thus process both rows and save the pairs (1,100) and (7,100) in their respective message caches.

When source 7 sends its own QUERY shortly afterward, it will carry the pair (7,100) as the (SENDER ID, SEQUENCE NUMBER) pair. It will be first received by neighbor nodes 3, 8 and 9. When nodes 8 and 9 receive this, they will check their message caches for the pair (7,100) and will find a match. They will hence consider this packet a duplicate and drop it. Node 3, another neighbor of source 7, will not find this pair in its message cache and will process it accordingly. Thus, in this example as well, there are three less QUERY transmissions than if there were no consolidation.

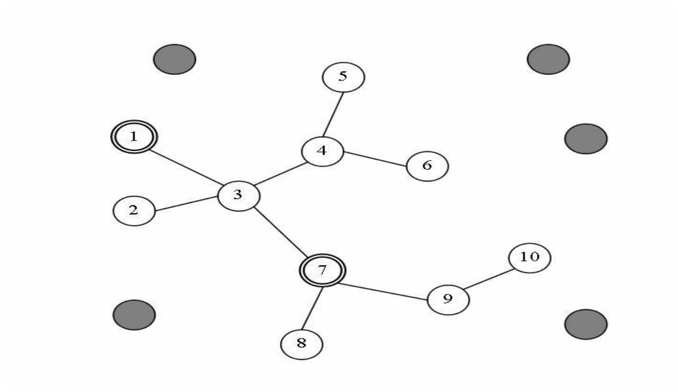

Fig. 3. Query consolidation at a source node

The choice of TIME-INTERVAL needs to be carefully made, as a large TIME-INTERVAL can lead to out-dated routes under high mobility, whereas a very small TIME-INTERVAL may not provide the full benefit of the protocol. In our simulations, we have used a value between one-fifth and one-tenth of the INT as the TIME-INTERVAL.

V. SIMULATION RESULTS

We evaluated the performance of our protocol against ODMRP by carrying out simulations in a GlomoSim environment. We also implemented DCMP and ODMRP-MPR. DCMP uses two parameters called MAXHOP and MAXPASSSIZE, which refer to maximum allowed distance between a CORE node and its PASSIVE node, and the maximum number of passive nodes allowed per core node, respectively. A source P can become a passive source for a core node C if C's MaxPassSize parameter is not exceeded, P is no more than MaxHop number of hops away from C, and C has a higher NodeID than P. If these conditions are met, P sends a PASSREQ packet toward C requesting to become its passive node, and C responds with a CONFIRM packet. P then refrains from sending its JOIN QUERY while it meets the conditions for its passive node status, and instead forwards its data packets to its core node. In ODMRP-MPR, each node periodically transmits a HELLO packet carrying the list of any neighbors that are known. Such HELLO packets allow a node to collect information about its 1-hop and 2-hop neighbor sets. This information is used to select a group of nodes called MultiPoint Relays (MPRs) [25] which would cover its entire 2-hop set. A known neighbor of a node X that is not in the MPR set of X does not need to transmit X's QUERIES.

The MAC protocol used in our simulations is IEEE 802.11 DCF. The simulation model consists of 100 mobile nodes within a 1500m x 1500m area. The nodes are placed randomly within this region. The mobility model used is random waypoint, in which each node independently picks a random destination and speed from an interval (*min*, *max*) and moves toward the chosen destination at this speed. Once it reaches the destination, it pauses for *pause* number of seconds and repeats the process. Our *min* speed is 1 m/s, *max* speed is 20 m/s and *pause* interval is 0 seconds. Thus, the nodes are constantly moving. A source generates data load at the rate of 5 Pkts/second. The QUERY interval is set at 3 second. The channel capacity is 2Mbps. Each simulation is run for 100 seconds. For each parameter combination, ten randomly generated scenarios are run and the results are averaged. For DCMP, the MaxHop is 3 and there is no limit on MaxPassSize. For ODMRP-MPR, the HELLO refresh interval is the same as the QUERY interval. In all other cases, we have used the same simulation parameters for all protocols unless othrwise specified.

A. Performance Metrics used:

- *Control Packet Load*: The average number of control packet tranmissions by a node in the network. Control packets include any of QUERY, REPLY, PASSREQ, CONFIRM, HELLO and Ack packets.
- *Packet delivery Ratio:* The ratio of data packets sent by all the sources that is received by a receiver.
- *Data packet Overhead:* The number of data transmissions performed by the protocol per successfully delivered data packet.
- *Control Packet Overhead:* The number of control transmissions performed by the protocol per successfully delivered data packet.
- *Total Packet Overhead:* The total of control and data overheads per successfully delivered data packet. This metric represents the multicast routing efficiency.

B. Results:

The Fig. 4 - 9 show the results of our simulation of ODMRP, CQMP, DCMP and ODMRP-MPR. DCMP performs better than ODMRP in most cases. This protocol may result in a weaker mesh between sources and receivers as the number of passive sources increases due to a reduction in the number of forwarding nodes. Additionally, a source that is passive may have a longer route to a receiver than if it were active. As a result, the number of data packets transmitted in the network may be larger than in ODMRP under a high number of sources. This is exhibited in Fig. 7. ODMRP-MPR can lead to varying results based on the order of arrival of HELLO packets. At its best, it would lead to a minimal set of multi-point relay (MPR) nodes and thus reduce the redundant transmission of QUERIES. Due to the flavor of the MPR heuristic ([23], [25]) used in this protocol, the worst case scenario, where all or most neighbors may be considered MPR nodes, also becomes a possibility. In this case, the HELLO packet overhead may actually lead to a higher overhead that ODMRP. This possibility is exhibited in our simulations in Fig. 4, 5 and 8.

CQMP, on the other hand, does not introduce any additional transmissions other than the ones defined in the ODMRP protocol. Since it works only with a QUERY already transmitted, it shows favorable results in all analyses.

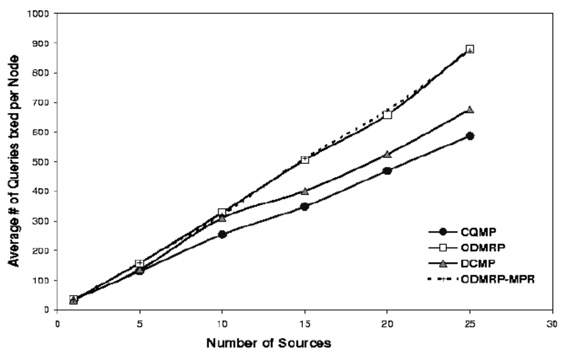

Fig. 4. Average number of Query Pkts. forwarded by a Node

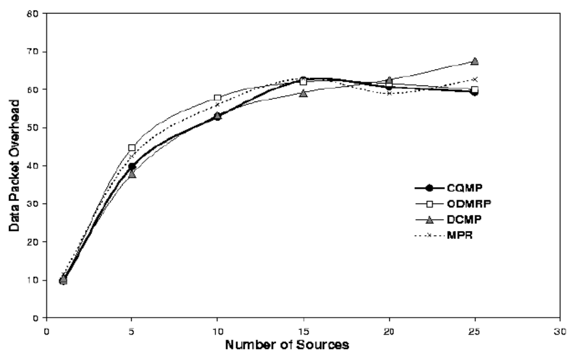

Fig. 7. Data Packet Overhead per Data Packet Delivered

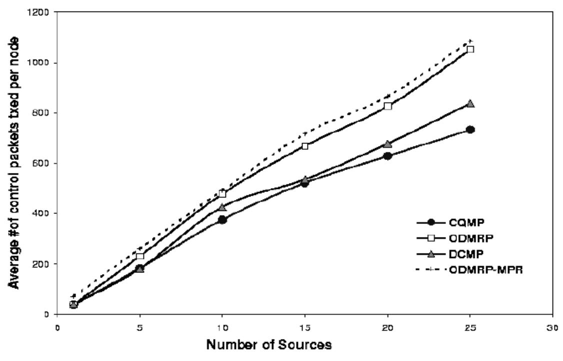

Fig. 5. Average number of Control Pkts. forwarded by a Node

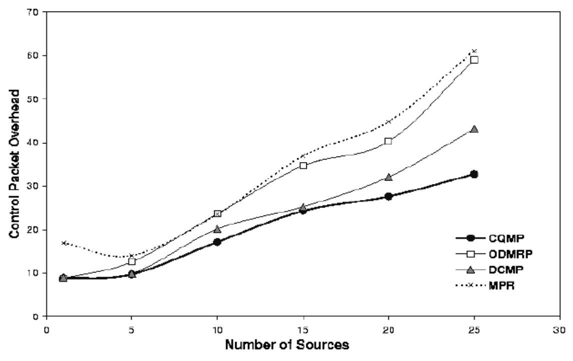

Fig. 8. Control Packet Overhead per Data Packet Delivered

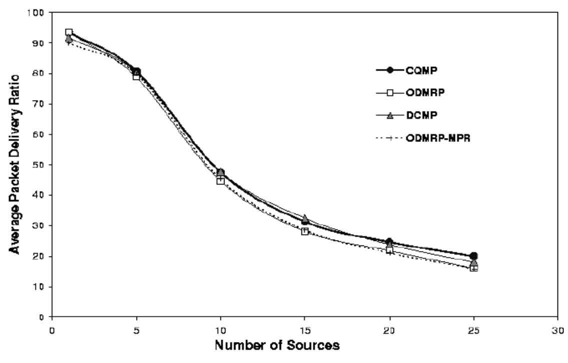

Fig. 6. Packet Delivery Ratio as a function of Number of Sources

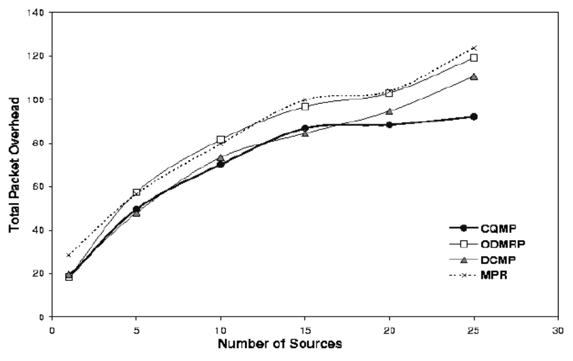

Fig. 9. Total Packet Overhead per Data Packet Delivered

The cause of high control overhead in ODMRP is redundant transmission of JOIN QUERY. Fig. 4 and 5 show the average number of QUERY packets transmitted by a node and the average number of all control packets transmitted by a node, respectively. CQMP performs better than the other protocols. The improvement of CQMP over ODMRP becomes more marked as the number of sources is increased. As the number of sources approaches 20 and beyond, CQMP produces almost 1/3 less average control packet load than that produced by ODMRP.

As shown by Fig. 6, the reduction in control packet load is also accompanied by an improvement in terms of data packet delivery ratio. Since CQMP has less QUERY packet transmissions than ODMRP, there is less chance of data packet loss by collision or congestion. The data delivery ratio of both ODMRP and CQMP decreases as the number of sources increases under high mobility conditions, but CQMP constantly maintains about 2 to 3 percent higher packet delivery ratio than ODMRP and consequently less data overhead as shown by Fig. 7.

Fig. 8 and 9 show the improvment of CQMP in terms of Control Packet Overhead and Total Data and Control Packet Overhead. CQMP performs better than all the compared protocols, especially as the number of sources increases. Our results thus show that by consolidating the QUERY at different levels, the control overhead of an ad-hoc network can be significantly reduced, while maintaining the desired level of packet delivery ratio.

,

VI. CONCLUSIONS

We have proposed a mesh-based, on-demand multicast routing protocol, CQMP, which uses consolidation of multicast group membership advertising packets. We implemented CQMP using GlomoSim and show by simulations that CQMP shows upto 30 percent reduction in control packet load and upto 20 percent improvement in multicast efficiency upon comparison with ODMRP. In addition, our results show that as the number of mobile sources increases, CQMP gives about 2 to 3 percent improvement over ODMRP in terms of data packet delivery ratio.

REFERENCES

[1] Anthony Ephremides, "Energy concerns in wireless networks," *IEEE Wireless Communications*, vol. 9, no. 4, pp. 48–59, Aug 2002.

[2] Charles E. Perkins, *Ad Hoc Networking*, Pearson Education, New Jersey, USA, Dec 2000.

[3] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proceedings of IEEE WMCSA'99*, Feb 1999, pp. 90–100.

[4] D. B. Johnson and D. A. Maltz, "The dynamic source routing protocol for mobile ad hoc networks," *Internet-Draft, draft-ietf-manet-dsr-02.txt*, 1999, Work in progress.

[5] Vincent D. Park and M. Scott Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*. 1997, pp. 1405–1413, IEEE.

[6] C. K. Toh, "Associativity based routing for ad hoc mobile networks," *Wireless Personal Communications*, vol. 4, no. 2, pp. 1–36, 1997.

[7] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi, "Signal stability based adaptive routing (ssa) for ad-hoc mobile networks," Tech. Rep., 1996.

[8] Z. J. Haas, M. R. Pearlman, and P. Samar, "Zone routing protocol (zrp)," *Internet-Draft, draft-ietf-manet-zpr-04.txt*, Jan 2001, Work in progress.

[9] Zygmunt J. Haas and Marc R. Pearlman, "The performance of query control schemes for the zone routing protocol," *IEEE/ACM Trans. Netw.*, vol. 9, no. 4, pp. 427–438, 2001.

[10] Samir Ranjan Das, Charles E. Perkins, and Elizabeth M. Belding-Royer, "Performance comparison of two on-demand routing protocols for ad hoc networks," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*. 2000, pp. 3–12, IEEE.

[11] Thomas Kunz, "Multicasting: from fixed networks to ad hoc networks," pp. 495–507, 2002.

[12] Jason Xie, Rajesh R. Talpade, Anthony McAuley, and Mingyan Liu, "AMRoute: Ad hoc multicast routing protocol," *MONET*, vol. 7, no. 6, pp. 429–439, 2002.

[13] C. Wu, Y. Tay, and C.-K. Toh, "Ad hoc multicast routing protocol utilizing increas ing id-numbers (amris) functional specification," *Internet-Draft, draft-ietf-manet-amris-spec-00.txt*, Nov 1998, Work in progress.

[14] Tomochika Ozaki, Jaime Bae Kim, and Tatsuya Suda, "Bandwidth-efficient multicast routing for multihop, ad-hoc wireless networks," in *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*. 2001, pp. 1182–1191, IEEE.

[15] Elizabeth M. Royer and Charles E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. 1999, pp. 207–218, ACM Press.

[16] P. Sinha, R. Sivakumar, and V. Bharghavan, "MCEDAR: Multicast core extraction distributed ad hoc routing," in *Proceedings of IEEE WCNC'99*, Sep 1999, pp. 1313–1317.

[17] William Su Sung-Ju Lee and Mario Gerla, "On-demand multicast routing protocol in multihop wireless mobile networks," in *Mobile Networks and Applications*. 2002, vol. 7, pp. 441–453, Kluwer Academic Publishers.

[18] Ching-Chuan Chiang, Mario Gerla, and Lixia Zhang, "Forwarding group multicast protocol (fgmp) for multihop, mobile wireless networks," *Cluster Computing*, vol. 1, no. 2, pp. 187–196, 1998.

[19] Ewerton L. Madruga and J. J. Garcia-Luna-Aceves, "Scalable multicasting: the core-assisted mesh protocol," *Mob. Netw. Appl.*, vol. 6, no. 2, pp. 151–165, 2001.

[20] Seungjoon Lee and Chongkwon Kim, "Neighbor supporting ad hoc multicast routing protocol," in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*. 2000, pp. 37–44, IEEE Press.

[21] Young-Bae Ko and Nitin H. Vaidya, "Geocasting in mobile ad hoc networks: Location-based multicast algorithms," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*. 1999, p. 101, IEEE Computer Society.

[22] B. S. Manoj Subir Kumar Das and C. Siva Ram Murthy, "A dynamic core based multicast routing protocol for ad hoc wireless networks," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, Lausanne, Switzerland*. 2002, pp. 24 – 35, ACM Press.

[23] Yao Zhao, Leiming Xu, and Meilin Shi, "On-demand multicast routing protocol with multipoint relay (odmrp-mpr) in mobile ad-hoc network," in *Proceedings of ICCT2003*. 2003, ICCT.

[24] Ken Tang and Mario Gerla, "Reliable on-demand multicast routing with congestion control in wireless ad-hoc networks," in *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*. 2003, IEEE.

[25] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol," *Internet-Draft, draft-ietf-manet-olsr-08.txt*, March 2003, Work in progress.

[26] Thomas Kunz and Ed Cheng, "Multicasting in ad-hoc networks: Comparing maodv and odmrp," in *Proceedings of the Workshop on Ad hoc Communications, Bonn, Germany*, 2001.

ACKNOWLEDGMENTS

Dr. Hung Q. Ngo is supported in part by NSF CAREER Award CCF-0347565 and by Telcordia Technologies Subcontract: FA8750-04-C-0249 from the DARPA SRS Program.