Volume 2, Issue 7, July 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

On Demand Multicast Routing Protocol for Mobile Adhoc Networks

G. Radhika
Department of CSE
M.Tech Student
Chadalawada Ramanamma Engg. college ,Tirupati
radi323@gmail.com

J.Ravi Kumar
Department of CSE
Assistant Professor
Chadalawada Ramanamma Engg. college Tirupati
ravikumar509@gmail.com

***Abstract*— In This paper presents a novel multicast routing protocol for mobile ad hoc wireless networks. The protocol, termed ODMRP (On-Demand Multicast Routing Protocol), is a mesh-based, rather than a conventional treebased, multicast scheme and uses a forwarding group concept (only a subset of nodes forwards the multicast packets via scoped flooding). It applies on-demand procedures to dynamically build routes and maintain multicast group membership. ODMRP is well suited for ad hoc wireless networks with mobile hosts where bandwidth is limited, topology changes frequently, and power is constrained. We evaluate ODMRP's scalability and performance via simulation.**

***Keywords*— Wireless Network, Multihop, Multicast, Ad hoc, Clustering, FGMP**

I. INTRODUCTION

Multicasting has emerged as one of the most focused areas in the field of networking. As the technology and popularity of the Internet have grown, applications that require multicasting (e.g., video conferencing) are becoming more widespread. Another interesting recent development has been the emergence of dynamically reconfigurable wireless ad hoc networks to interconnect mobile users for applications ranging from disaster recovery to distributed collaborative computing.
Multicast tree structures are fragile and must be readjusted continuously as connectivity changes. Furthermore, typical multicast trees usually require a global routing substructure such as link state or distance vector. The frequent exchange of routing vectors or link state tables, triggered by continuous topology changes, yields excessive channel and processing overhead. Limited bandwidth, constrained power, and mobility of network hosts make the multicast protocol design particularly challenging.
To overcome these limitations, we have developed the On-Demand Multicast Routing Protocol (ODMRP). ODMRP applies *on-demand* routing techniques to avoid channel overhead and improve scalability. It uses the concept of *forwarding group* [5], a set of nodes responsible for forwarding multicast data on shortest paths between any member pairs, to build a forwarding *mesh* for each multicast group. By and using a mesh instead of a tree, the drawbacks of multicast trees in mobile wireless networks (e.g., intermittent connectivity, traffic concentration, frequent tree reconfiguration, non-shortest path in a shared tree, etc.) are avoided. A *softstate* approach is taken in ODMRP to maintain multicast group members. No explicit control message is required to leave the group. We believe the reduction of channel/storage overhead and the relaxed connectivity make ODMRP more scalable for large networks and more stable for mobile wireless networks.

II. OVER VIEW OF ON DEMAND MULTICAST PROTOCOL

A. Maintain membership and multicast routing:

On Demand Multicast routing is Similar to on-demand unicast routing protocols, a request phase and a reply phase comprise the protocol (see Fig. 1).While a multicast source has packets to send, it periodically broadcasts to the entire network a member advertising packet, called a JOIN REQUEST. This periodic transmission refreshes the membership information and updates the route as follows. When a node receives a non-duplicate JOIN REQUEST, it stores the upstream node ID (i.e., backward learning) and rebroadcasts the packet. When the JOIN REQUEST packet reaches a multicast receiver, the receiver creates or updates the source entry in its Member Table. While valid entries exist in the Member Table, JOIN TABLES are broadcasted periodically to the neighbors. When a node receives a JOIN TABLE, it checks if the next node ID of one of the entries matches its own ID. If it does, the node realizes that it is on the path to the source and thus is part of the forwarding group. It then sets the FG Flag and broadcasts its own JOIN TABLE built upon matched entries. The JOIN TABLE is thus propagated by each forwarding group member until it reaches the multicast source via the shortest path. This

process constructs (or updates) the routes from sources to receivers and builds a mesh of nodes, the *forwarding group*.

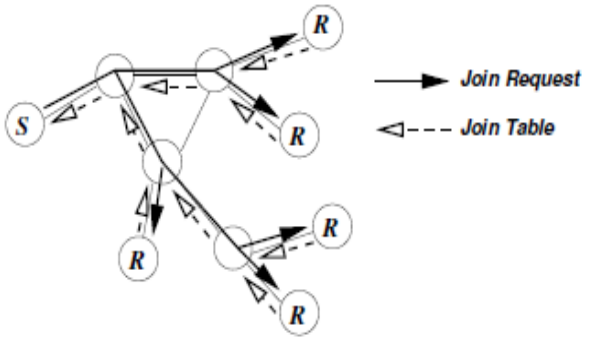


Fig. 1. On-Demand Procedure for Membership Setup and Maintenance.

We have visualized the forwarding group concept in Fig. 2. The forwarding group is a set of nodes in charge of forwarding multicast packets. It supports shortest paths between any member pairs. All nodes inside the .bubble. (multicast members and forwarding group nodes) forward multicast data packets. Note that a multicast receiver can also be a forwarding group node if it is on the path between a multicast source and another receiver. The mesh provides richer connectivity among multicast members compared to trees. Flooding redundancy among forwarding group helps overcome node displacements and channel fading. Hence, unlike trees, frequent reconfigurations are not required.

Fig. 3 is an example to show the robustness of a mesh configuration. Three sources (S1,S2 ,and S3) send multicast data packets to three receivers (R1,R2,and R3) via three forwarding group nodes (A,B, and C). Suppose S1 to R2 is S1-A-B-R2. In a tree configuration, if the link between nodes A and B breaks or fails,R2 cannot receive any packets from S1until the tree is reconfigured. ODMRP, on the other hand, already has a redundant route (e.g., S1-A-C-B-R2) to deliver packets without going through the broken link between nodes A and B .

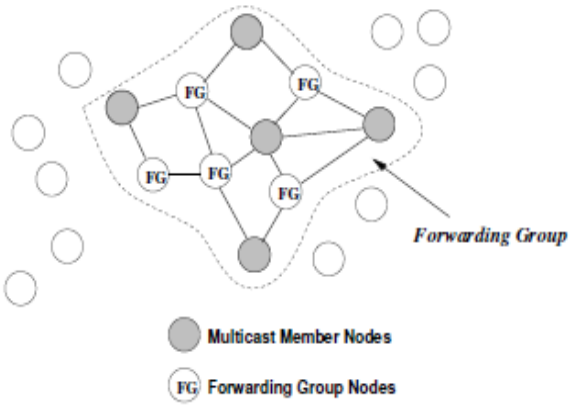


Fig. 2. The Forwarding Group Concept.

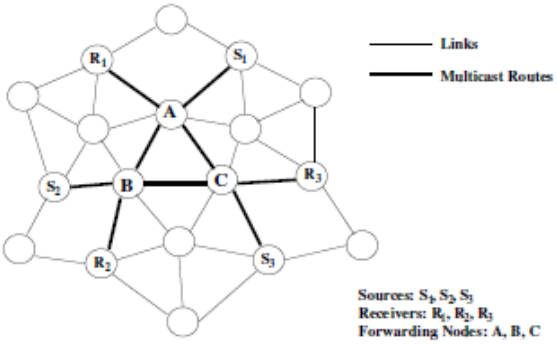


Fig. 3. Why a Mesh?

B. Data Forwarding

After the group establishment and route construction process, a multicast source can transmit packets to receivers via selected routes and forwarding groups. Periodic control packets are sent only when outgoing data packets are still present.

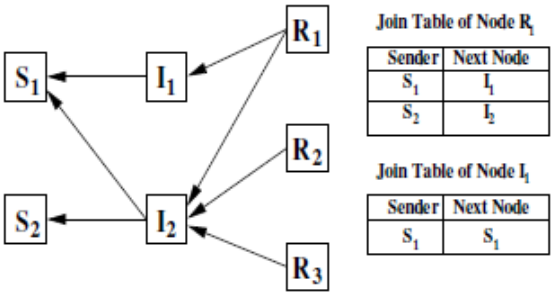


Fig. 4. An Example of a Join Table Forwarding.

When receiving a multicast data packet, a node forwards it only if it is not a duplicate and the setting of the FG-Flag for the multicast group has not expired. This procedure minimizes traffic overhead and prevents sending packets through stale routes.

C.Soft State

In ODMRP, no explicit control packets need to be sent tojoin or leave the group. If a multicast source wants to leave the group, it simply stops sending JOIN REQUEST packets since it does not have any multicast data to send to the group. If a receiver no longer wants to receive from a particular multicast group, it removes the corresponding entries from its Member Table and does not transmit the JOIN TABLE for that group. Nodes in the forwarding group are demoted to non-forwarding nodes if not refreshed (no JOIN TABLES received) before they timeout.

D. Data Structures

Network hosts runningODMRP are required to maintain the following data structures.
Member Table :

Each multicast receiver stores the source information in the Member Table. For each multicast group the node is participating in, the source ID and the time when the last JOIN REQUEST is received from the source is recorded. If no JOIN REQUEST is received from a source within the refresh period, that entry is removed from the Member Table.

Routing Table : ARouting Table is created on demand and is maintained by each node. An entry is inserted or updated when a non-duplicate JOIN REQUEST is received. The node stores the destination (i.e., the source of the JOIN REQUEST) and the next hop to the destination (i.e., the last node that propagated the JOIN REQUEST). The Routing Table provides the next hop information when transmitting Join Tables.

Forwarding Group Table : When a node is a forwarding group node of the multicast group, it maintains the group information in the Forwarding Group Table. The multicast group ID and the time when the node was last refreshed is recorded.

Message Cache: The Message Cache is maintained by each node to detect duplicates. When a node receives a new JOIN REQUEST or data, it stores the source ID and the sequence number of the packet. Note that entries in the Message Cache need not be maintained permanently. Schemes such as LRU (Least Recently Used) or FIFO (First In First Out) can be employed to expire and remove old entries and prevent the size of the Message Cache to be extensive.

E. Unicast Capability

One of the major strengths of ODMRP is its unicast routing capability. Not only ODMRP can work with any unicast routing protocol, it can function as both multicast and unicast. Thus, ODMRP can run without any underlying unicast protocol. Other ad hoc multicast routing protocols such as AMRoute [3], CAMP [9], RBM [6], and LAM [11] must be run on top of a unicast routing protocol. CAMP, RBM, and LAM in particular, only work on top of certain underlying unicast protocols.

III. PERFORMANCE EVALUATION

A. *Simulation Environment*

The simulator is implemented within the Global Mobile Simulation (GloMoSim) library [18]. The GloMoSim library is a scalable simulation environment for wireless network systems using the parallel discrete-event simulation capability provided by PARSEC [1]. The simulation of parallel programs typically suffers from slowdown factors of 30 to 50 *per processor*.[1] This means that simulation of a parallel program executing for five to 15 minutes on a 128-node multiprocessor can take anywhere from a few weeks to a few months—even on state-of-the-art sequential workstations. As the size of the physical system increases,the models' memory requirements can easily exceed the workstations' capacity.

The limitations of sequential model execution have led to growing interest in the use of parallel execution for simulating large-scale systems. Widespread use of parallel simulation, however, has been significantly hindered by a lack of tools for integrating parallel model execution into the overall framework of system simulation. Although a number of algorithmic alternatives exist for parallel execution of discreteevent simulation models, performance analysts not expert in parallel simulation have relatively few tools giving them flexibility to experiment with multiple algorithmic or architectural alternatives for model execution. Another drawback to widespread use of simulations is the cost of model design and maintenance. The design and development costs for detailed simulation models for complex systems can easily rival the costs= for the physical systems themselves.The simulation environment we developed at UCLA attempts to address some of these issues by providing these features:

• An easy path for the migration of simulation models to operational software prototypes.
• Implementation on both distributed- and sharedmemory platforms and support for a diverse set of parallel simulation protocols.
• Support for visual and hierarchical model design.

Our environment consists of three primary components: a parallel simulation language called Parsec (parallel simulation environment for complex systems); its GUI, called Pave; and the portable runtime system that implements the simulation algorithms.

Parsec is based on the Maisie simulation language, with these significant modifications:

• It uses simpler syntax.
• It uses modified language, to facilitate porting code from the simulation model to the operational software.
• It has a robust and extensible runtime kernel that is considerably more efficient than its predecessor.
• It uses new protocols to predict parallel performance.

Parsec

Parsec adopts the process-interaction approach to discrete-event simulation. A Parsec program consists of a set of *entities* and C functions. Each entity is an LP that models a corresponding physical process; entities can be created and destroyed dynamically. Events are modeled by message communications among the corresponding entities. Each message carries a logical time stamp matching the time at which the corresponding event occurs in the physical system. An entity may also schedule for itself a special message, called a *timeout*, for a specific time in the future. This message is often used by an entity to simulate the passage of time in the physical system, and its handling has been optimized in the system.

Wireless networking.

The following code fragment is a Parsec entity for simulating part of a wireless net working protocol. The entity implements the media access control (MAC) layer, which takes packets from a network-routing protocol and transmits them with a radio.

```
message  Cleared ToSend { };
```

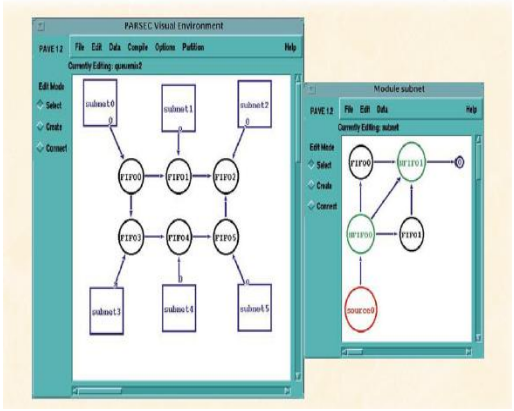


```
message MACPacket {IP dest; byte buffer[ PACKET_SIZE];};
message  NetworkPacket {IP dest; byte buffer[PACKET_SIZE];};
message  RadioPacket {IP dest; byte buffer[PACKET_SIZE];};
message  RequestClearToSend {IP dest;};
entity  MAC_Protocol (IP myIP) {
bool transmitting;
ename  radio, network;
message  MACPacket macPacket;
while (true) {
receive (NetworkPacket np) when
(!transmitting) {
transmitting = true;
push (np, buffer);
send RequestClear To Send
{np.dest} to radio;
}
or receive (ClearedToSend cts) {
macPacket =
buildPacket(pop(buffer));
send macPacket to radio;
hold(TRANSMISSION_DELAY);
transmitting = false;
}
or receive {RadioPacket rp) {
send rp to network;
}
}
}
```

First, several message types are defined just like a C **struct** code declaration. Then the entity itself is defined, much like a C function. The body of the entity consists primarily of a receive statement block, which has several *resume clauses*. The first resume clause accepts a packet from the network routing layer, but the when clause prevents this message from being accepted
while the entity is transmitting an earlier packet. When the entity does receive a packet, it instructs the radio to request a transmission window (Request ClearToSend), buffers the packet, and changes its state to "transmitting" until the packet is sent. The packet must be buffered because the variable *np* is local to the resume clause where it is declared. When the entity receives the clear signal from the radio, it delivers the packet and uses the hold statement to advance its time to after the transmission. The MAC layer may also receive the radio packets, which it forwards to the network routing layer. Notice that there are two forms
of the send statement: one which uses the message type and a parameter list and a simpler one that just sends a message variable.

Visual model design

The Parsec Visual Environment (Pave) facilitates the visual design of related simulation component libraries, the construction of simulation models from these components in a simple visual framework, the generation of Parsec code for the models, and the optimization of the models for parallel execution. Component libraries can be created for domains such as queuing networks or mobile network infrastructures. Unlike most existing visual simulation tools, Pave was designed specifically to support parallel simulations.



The metrics used in ODMRP evaluation are:

Packet Delivery Ratio: The number of data packet delivered to multicast receivers over the number of data packets supposed to be delivered to multicast receivers.
Number of Control Bytes Transmitted per Data Byte Delivered: Instead of using a pure control overhead, we choose to use a ratio of control bytes transmitted to data byte delivered to investigate how efficiently control packets are utilized in delivering data. In addition to bytes of control packets (e.g., JOIN REQUESTS, JOIN TABLES), bytes of data packet headers are included in calculating control bytes transmitted. Accordingly, only bytes of the data payload contributes to the data bytes delivered.
Number of Data and Control Packets Transmitted per Data Packet Delivered: This measure shows the efficiency in terms of channel access and is very important in ad hoc networks since link layer protocols are typically contention-based.

B. *Simulation Results*

Fig. 5 shows the packet delivery ratio of ODMRP as a function of mobility speed. The size of multicast group is varied to examine the scalability of the protocol. Having only two multicast members corresponds to a unicast situation. The result indicates that ODMRP delivers high portion of data packets in most of our scenarios. In highly mobile situations, the performance is the least effective in the two members case. When ODMRP functions as a unicast protocol, a mesh is not formed and there is no redundancy in packet forwarding. Since there are no multiple routes, the probability of packet drop increases with mobility speed.
The average number of control bytes transmitted per data byte delivered is shown in Fig. 6. We can see that ODMRP efficiently utilizes control packets in delivering data. JOIN REQUESTS are transmitted by the source only when it has data to send. JOIN TABLES are sent by receivers when valid sources exist in their Member Table. Thus, control packets are generated only if needed and all the control messages are utilized in establishing or refreshing routes and group membership. Furthermore, the transmission of control packets

is periodic and the pure control overhead remains relatively constant regardless of mobility speed.

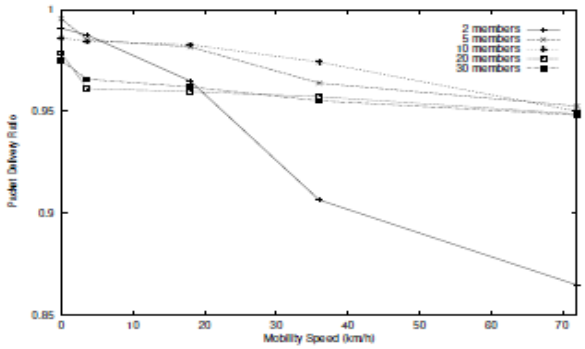


Fig. 5. Packet Delivery Ratio.

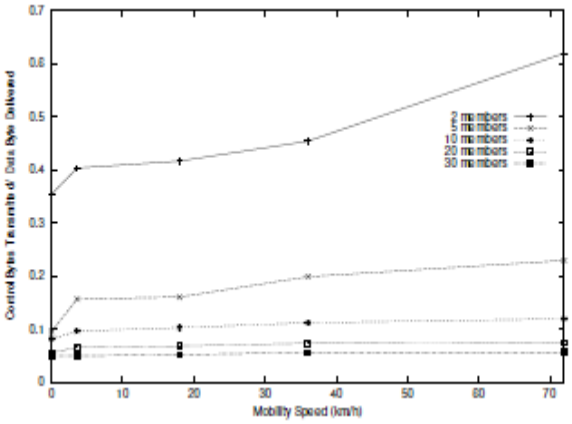


Fig. 6. Number of Control Bytes Transmitted per Data Byte Delivered.

IV. RELATED WORKS

Other multicasting protocols have been proposed for ad hoc networks. The Reservation-Based Multicast (RBM) routing protocol [6] builds a core (or a Rendezvous Point) based tree for each multicast group. RBM is a combination of multicast, resource reservation, and admission control protocol where users specify requirements and constraints. The Lightweight Adaptive Multicast (LAM) algorithm [11] is a group shared tree protocol that does not require timer-based messaging. Similar to other core-based protocols, it suffers from disadvantages of traffic concentration and vulnerability of the core. The Adhoc Multicast Routing Protocol (AMRoute) [3] is also a shared-tree protocol which allows dynamic core migration based on group membership and network configuration.

V. CONCLUSIONS

We have proposed ODMRP (On-Demand Multicast Routing Protocol) for a mobile ad hoc wireless network. ODMRP is based on mesh (instead of tree) forwarding. It applies ondemand (as opposed to periodic) multicast route construction and membership maintenance. Simulation results show that ODMRP is effective and efficient in dynamic environments and scales well to a large number of multicast members. The advantages of ODMRP are:

- Low channel and storage overhead
- Usage of up-to-date and shortest routes
- Robustness to host mobility
- Maintenance and exploitation of multiple redundant paths
- Scalability to a large number of nodes
- Exploitation of the broadcast nature of wireless environments
- Unicast routing capability

Various improvements of the protocol are in progress and will be reported in an upcoming paper.

REFERENCES

[1] M.S. Corson and S.G. Batsell, .A Reservation-Based Multicast (RBM) Routing Protocol for Mobile Networks: Initial Route Construction Phase,. *ACM/Baltzer Wireless Networks*, vol. 1, no. 4, Dec. 1995, pp. 427-450.

[2] E.A. Brewer et al., *Proteus: A High Performance Parallel Architecture Simulator*, Tech. Report MIT/LCS/ TR- 516, Massachusetts Institute of Technology, Cambridge, Mass., 1991.

[3] L. Ji and M.S. Corson, .A Lightweight Adaptive Multicast Algorithm,. In *Proceedings of IEEE GLOBECOM'98*, Sydney, Australia, Nov. 1998, pp. 1036-1042.

[4] P. Johansson, T. Larsson, N. Hedman, and B. Mielczarek, .Routing Protocols for Mobile Ad-hoc Networks – A Comparative Performance Analysis, . To appear in *Proceedings of ACM/IEEE MOBICOM'99*, Seattle, WA, Aug. 1999.

[5] UCLA Parallel Computing Laboratory and Wireless Adaptive Mobility Laboratory, GloMoSim: A Scalable Simulation Environment for Wireless and Wired Network Systems.

[6] C. E. Perkins and E. M. Royer. Ad-hoc On- Demand Distance Vector Routing. Proceedings of the 2”d IEEE Workshop on Mobile Computing Systems and Applications, pages 90-100, New Orleans, LA, February 1999.

[7] R. Bagrodia and W. Liao. Maisie: A Language for Design of Efficient Discrete Event Simulation. IEEE Transactions on Software Engineering, April 1994.

[8] R. Caceres and V. N. Padmanabhan. Fast and Scalable Handoffs for Wireless Internetworks. Proceedings of the 2nd ACM International Conference on Mobile Computing and Networking, pages 56-66, November 1996.

[9] H. Eriksson. MBONE: The Multicast Backbone. Communications of the ACM, 37(8):54-60, August 1994.

[10] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. IEEE/ACM Transactions on Networking,5(6):784-803, December 1997.

