Multicast Protocol Implementation and Validation in an Ad hoc Network Testbed

Sang Ho Bae*, Sung-Ju Lee†, and Mario Gerla*
* Computer Science Department, University of California, Los Angeles
† Internet & Mobile Systems Labs, Hewlett-Packard Laboratories

http://www.cs.ucla.edu/NRL/wireless

Abstract* – *We present our experiences in implementing and validating the On-Demand Multicast Routing Protocol (ODMRP) in a real wireless ad hoc network testbed. ODMRP maintains a mesh for each multicast group to provide multiple alternate paths. Redundancy created by the mesh helps overcome frequent topology changes resulting from node mobility, channel fading, and interferences. The protocol does not maintain permanent route tables with full topological views. Instead, multicast senders reactively and dynamically discover routes and obtain multicast group information on demand. ODMRP is implemented in our testbed network consisting of six hosts using the kernel level multicast support option built into the Linux operating system. We describe the key design and implementation features of our protocol and report preliminary testbed experiment results of ODMRP and DVMRP (Distance Vector Multicast Routing Protocol), a traditional tree based scheme.

I. INTRODUCTION

Today's computing needs demand the support of network infrastructure, but in the event of natural catastrophe, war, or geographic isolation, network connectivity is not always attainable. Ad hoc networks provide rapidly deployable mobile infrastructures to fill such voids in communication links. Mobility of ad hoc networks introduces several constraints in protocol design. The routes of ad hoc networks are often "multihop" because of the limited radio propagation range of wireless devices. The routing schemes must handle frequent topology changes resulting from node mobility, channel fading, and interference. The nodes within the ad hoc network need to contend for the limited bandwidth and conserve battery power. These constraints make the routing and multicasting in ad hoc networks challenging.

Multicasting allows network hosts to transmit voice, video, and/or text to multiple destinations simultaneously without repetitious transmissions. In a typical ad hoc environment, network hosts work in groups to carry out a given task. Multicast hence plays an important role. The traditional multicast methods proposed for wired infrastructures such as DVMRP [3], MOSPF [11], CBT [2], and PIM [4] are not well suited to handle the mobility in wireless ad hoc networks. These schemes 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.

This work was funded in part by Intel and the Defense Advanced Research Projects Agency (DARPA) under contract DAAB07-97-C-D321, as a part of the Global Mobile Information Systems (GloMo) program.

To overcome these limitations, we developed the On-Demand Multicast Routing Protocol (ODMRP) [1], [8]. ODMRP applies an on-demand routing technique to avoid channel overhead and improve scalability. It uses the concept of the forwarding group, 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 maintaining and using a mesh, the drawbacks of multicast trees in mobile wireless networks (intermittent connectivity, traffic concentration, frequent tree reconfiguration, non-shortest path in a shared tree) are avoided. ODMRP takes a soft-state approach to maintain multicast group members. Nodes need not send any explicit control message to leave the group.

ODMRP and its performance have been studied in extensive simulations, and it is shown to outperform other ad hoc multicast protocols [10]. Our next step in protocol development is to test the protocol in real networks. We implemented ODMRP in the Linux operating system by utilizing the multicast extension built into the kernel. In [1], we demonstrated the overhead caused by DVMRP while forwarding the MBone traffic through wired-to-wireless gateway in a four node testbed and presented the initial local operation of ODMRP in the same testbed. In this paper, we extend our previous work by increasing the testbed complexity and making a direct performance comparison between ODMRP and DVMRP by localizing both multicast protocols in the same network environment. Our goal of the experiments is to validate the correctness of our protocol.

The rest of this paper is organized as follows. ODMRP is briefly overviewed in Section II followed by the protocol implementation description in Section III. Performance evaluation results and analysis are discussed in Section IV, and concluding remarks are made in Section V.

II. ODMRP OVERVIEW

This section gives a short overview of the On-Demand Multicast Routing Protocol (ODMRP). For a detailed operation of the protocol, readers are referred to [8], [9].

ODMRP establishes and updates group membership and multicast routes by the source on demand. Similar to on-demand unicast routing protocols, a query phase and a reply phase comprise the protocol (see Figure 1). While a multicast source has packets to send, it floods a member advertising

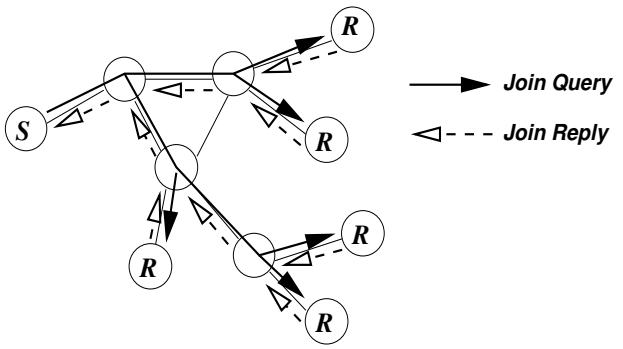

Fig. 1. On-demand procedure for membership setup and maintenance.

packet with data payload attached. This packet, called JOIN QUERY, is periodically broadcasted to the entire network to refresh the membership information and update the routes. When a node receives a non-duplicate JOIN QUERY, it stores the upstream node address (i.e., backward learning) into the route table and rebroadcasts the packet. When the JOIN QUERY packet reaches a multicast receiver, the receiver creates and broadcasts a JOIN REPLY to its neighbors. This JOIN REPLY packet is propagated all the way back to the source, and nodes in between become part of the route and "the forwarding group."

After the group establishment and route construction process, a source can multicast packets to receivers via selected routes and forwarding groups. When receiving the multicast data packet, a node forwards the packet only when it is not a duplicate and the node is a part of the forwarding group of the multicast session. This procedure minimizes the traffic overhead and prevents sending packets through stale routes while still providing alternate routes and richer connectivity between multicast members.

III. IMPLEMENTATION

A. Implementation Platform

A.1 Operating System and Software

We developed ODMRP on Linux kernel version 2.0.36, the version provided by the Red Hat Linux version 5.2. All tools and software packages that we used in our development originate from software bundle incorporated within the Red Hat Linux version 5.2 operating system package with an exception of Lucent WaveLan IEEE 802.11 device driver [13]. We chose the Linux operating system for its availability, familiarity, and most importantly, kernel level support for multicasting. The kernel support for multicast allows fast kernel level multicast packet switching and minimizes expensive delays caused by kernel-to-application and application-to-kernel level crossing.

In Section IV, we study the bandwidth utilization of DVMRP in a wireless environment by routing the Multicast Backbone (MBone) [5] traffic from wired to wireless network with the Linux version of `mrouted`, a DVMRP based multicast routing daemon.

A.2 Hardware

Ad hoc network nodes consist of Intel Pentium II based Hewlett Packard Omnibook 7150 laptops and Texas Instruments Extensa 510 laptops equipped with Lucent IEEE 802.11 WaveLan radio devices. The WaveLan devices operate on 2.4 GHz bandwidth and communicate at the maximum capacity of 2 Mb/s with the semi-open space range of 150 meters. The WaveLan devices are operated in an ad hoc mode.

B. Software Architecture

ODMRP uses the kernel level multicast support option built into the Linux operating system. With the exception of a minor alteration made to allow single device forwarding, we did not make any changes at the kernel level. The Linux kernel supports multicast by performing the following procedures. The user enables the multicast option in the network interface driver. The multicast enabled interface accepts and sends all packets with multicast address to the kernel. The kernel accepts all multicast packets, stores them in the message cache, and starts the cache timer. The cached messages are discarded when the timer expires. The kernel then periodically looks for the cached group addresses in the kernel multicast route table and decides whether to forward the cached messages or not. The messages are forwarded by altering their forwarding destination interfaces and buffering them to the corresponding interfaces. The messages are sent up to the user level only if there exists a local multicast application that has joined the group. This process avoids the costly kernel-to-user crossing for store-and-forward packets and improves efficiency. The destination interface is changed in accordance with the listings in the kernel level multicast route table. The kernel level route table is updated and maintained by a user level routing daemon which keeps a local image of the kernel level route table. The user level table is copied to the kernel level table as soon as updates are made. We used this basic route table interface to build and maintain "mirrored" ODMRP route tables both at the kernel and the user level. In the following sections, we describe our schemes to manage the control packets and the route table and discuss a forwarding scheme based on virtual interfaces.

B.1 Packet and Table Management

There are two types of control packets in ODMRP. JOIN QUERY packets advertise the multicast session and JOIN REPLY packets establish the path and the forwarding group. We implemented these packets as new types of Internet Group Management Protocol (IGMP) [6] packet which includes a data section. We expanded existing IGMP packet structure and handler function to include JOIN QUERY and JOIN REPLY functionalities.

When a JOIN QUERY packet arrives at the router, the router caches the content of the packet into a temporary route table (`tr_table`) and starts the timer for the entry. If the router does not receive a corresponding JOIN REPLY in time, the timer expires and the router removes the cached entry. If a JOIN REPLY which has a corresponding entry in the `tr_table` arrives before the timeout, the user level route table (`route_table`) is searched to find the <source, multicast group> pair that matches the `tr_table` entry. If such a pair is found, the router resets the soft state timer for the entry and waits for the next event. If the pair can not be found in the `route_table`, the router creates and inserts a new entry into the table. Routers periodically check the `route_table` timer expiration and remove expired entries. Whenever an entry is inserted or deleted, the router activates the trigger for the update of the kernel level route table (`kr_table`).

B.2 Forwarding on Virtual Interfaces

The DVMRP, PIM, and CBT based multicast routers are all built to be used over wired networks. Therefore, their frameworks are designed for routers with multiple network interfaces. The forwarding capability of these systems is limited strictly to passing the packets from one interface to another. In wired networks, having multiple interfaces does not cause any problems since the devices do not interfere with one another. This is not the case with our wireless network testbed because we use omni-directional antennas and a common broadcast channel. Having multiple wireless interfaces does not improve the performance. Unless specifically configured (for example, at different frequencies), the devices interfere with one another. The framework in Linux however, allows the forwarding between virtual interfaces (VIF) to support the tunneling among the multicast islands. A virtual interface can be created on a physical device in two ways. An IP alias can be created on a physical device. In Linux version 2.0.36, we can create an IP alias by adding an interface entry with a new IP address and a network device alias onto the kernel interface configuration table. We can then use this interface in exactly the same manner as the original physical device with all its physical attributes. We can create a virtual interface by opening a tunnel between two multicast routers. Unlike VIF that is created with the aliasing method, only the multicast router can use a tunnel since the multicast routing daemon establishes the tunnel by opening a unicast socket to encapsulate the multicast streams. In our experiments, we make single device forwarding possible by aliasing the existing hardware interface to create VIFs and then enabling the forwarding of the multicast packets to VIFs corresponding to the `route_table` entries.

B.3 ODMRP Timers

For the ODMRP soft state timer values, we selected one second for route refresh interval and five seconds for forwarding group timeout interval. The source of active sessions periodically refreshes the multicast route every *route refresh interval*. The forwarding group nodes are expired and demoted to non-FG nodes when they are not refreshed (not receive a JOIN REPLY) within the *forwarding group timeout interval*.

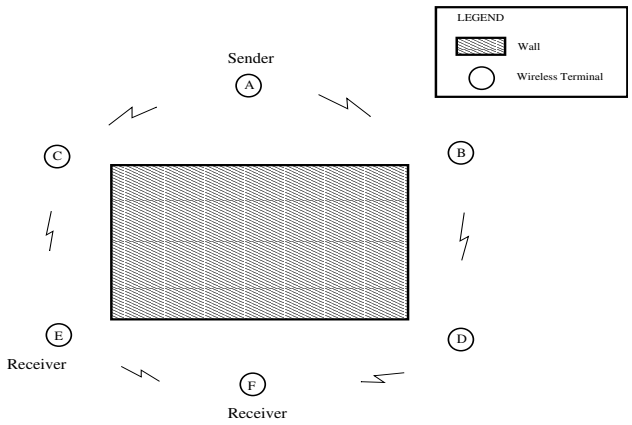

Fig. 2. Our testbed topology.

IV. PERFORMANCE EVALUATION

We created a six node testbed for our multicast experiments. We study the bandwidth utilizations of ODMRP and DVMRP. We intended to compare ODMRP with other ad hoc wireless multicast protocols, but no multicast testbed implementation is released to the public for testing. We were able to obtain the DVMRP implementation [12], and compare its performance with ODMRP in our study. The experimental setting is shown in Figure 2. Our network consists of six nodes with the node A serving as the gateway for the wireless network in the DVMRP experiment and as a multicast source in the ODMRP exprriment. Nodes E and F are receivers. The MBone experiment of DVMRP is performed by introducing the wired multicast traffic onto the wireless network through node A. We used the Linux DVMRP implementation of the `mrouted` version 3.81 developed at Stanford University. We present and analyze the results of ODMRP and DVMRP in this section.

A. DVMRP Overview

DVMRP (Distance Vector Multicast Routing Protocol) is based on the distance-vector routing algorithm. In this protocol, each router maintains a route table with all reachable destinations. A typical route table entry consists of destination address, metric to the destination (such as distance or hop count), and the next hop to reach the destination. The router obtains the up-to-date routing information by periodically exchanging the route table with immediate neighbors. After each exchange, routers compute shortest paths and update new information to the route table. To accommodate the multicast, a route entry includes the multicast group address, children

TABLE I

DVMRP WITH MBONE FEED.

	Value	% of total
Avg. session length	178 sec	N/A
IGMP control packet O/H	15.58 kb/s	6 %
Avg. # of active multicast channels	2	N/A
O/H caused by multicast channels	26.41 kb/s	10.29 %
Effective data throughput	214.71 kb/s	83.71 %
Total throughput (data and control)	256.7 kb/s	100 %

TABLE II

DVMRP WITH A LOCAL SOURCE.

	Value	% of total
Avg. session length	N/A	N/A
IGMP Control packet O/H	0.79 kb/s	0.13 %
Avg. # of active multicast channels	1	N/A
O/H caused by multicast channels	0 kb/s	0 %
Effective data throughput	608.5 kb/s	99.87%
Total throughput (data and control)	609.29 kb/s	100 %

routers' membership information, and the local subnet membership information fields. The routing information distributed among the routers collectively creates a multicast tree for each multicast group. When a new router joins the network, the multicast streams and the route table of neighboring nodes are forwarded to the new router. To leave an unwanted multicast session, the router must send a prune message. A node is required to join the group if there exists a member among its descendent nodes. DVMRP relies on IGMP to request the routing information and to exchange the control messages. IGMP control messages are also used for probing the neighboring routers for the active status of the next hop multicast daemon.

B. DVMRP Trace Analysis with MBone Sources

The multicast routing daemon `mrouted` is installed in each node. The base station, marked as a sender, links wired and wireless networks. The multicast daemons are activated one by one in each node. The time it takes for the routing daemon to stabilize as the route updates from the parent node in the multicast tree are forwarded is variable. The duration of time depends on the number of <sender, destination> pairs in the network. We do not include the overhead required by the DVMRP initialization in the analysis because we cannot consider the initialization period as a part of the normal operation mode. The testbed is ready for measurement experiments only after the initial control packet rush subsides and a regular control packet traffic pattern emerges. We carry out the experiment in the following steps. Each router starts the traffic traces using `tcpdump`. The receivers at nodes E and F then join audio and video multicast sessions. We monitored the multicast for approximately three minutes. Table I reports the results.

DVMRP operates by first allowing all multicast streams to be forwarded downstream and then selectively "pruning" the unwanted streams bottom-up with IGMP messages. This practice works well in wired networks since very few control packets are lost. In wireless networks however, packet loss is frequent. When a prune message is lost at a router for instance, the corresponding multicast group is allowed to continue forwarding from the router down until the next prune cycle. This unnecessary forwarding causes the channel overhead listed in Table I. If a more complex topology had been deployed, there would be even more control packet losses, with higher bandwidth wastage.

C. DVMRP Trace Analysis with a Local Source

For this experiment, the base station that bridged the wired and wireless networks in the MBone experiment is replaced with a wireless multicast source. We kept the rest of the topology the same. The multicast stream consists of a file multicast from the sender node A to the receivers E and F. Table II shows the measurement results. The sessions, in this experiment, last only until the file is transfered, so the session length field in the table is left blank. Comparing with the results in Table I, we note a much less control packet overhead. Recall that the DVMRP control overhead per node increases in proportion to the number of <source, multicast group> pairs in the network. Even though only two multicast streams were active in the MBone experiment, the multicast route table for all active <source, group> pairs was forwarded through control packets and caused high overhead. In fact, the multicast route table refresh requires one IGMP poll per multicast group and per source in the group. In the local DVMRP experiment however, the control message overhead is much lower since there is no external multicast group pair. There is also an increase in the total throughput, but this result is not an indication of the performance improvement. The two multicast channels (streams) that are propagated to the wireless network in the MBone experiment are source rate limited to conserve bandwidth. Our result reflects the source limiting.

D. ODMRP Trace Analysis

In the ODMRP experiments, we kept the topology the same as in the DVMRP experiments. The ODMRP multicast routing daemon does not need the stabilizing period required by the DVMRP since no control packet is switched between routers to establish the initial state. Currently, the MBone traffic cannot be forwarded onto the ODMRP testbed, so we replicated only the single source multicasting of the Section IV-C experiment. Table III shows the results.

By comparing Table II with Table III, we note that neither DVMRP or ODMRP experiments achieve the full WaveLAN

TABLE III

ODMRP WITH A LOCAL SOURCE.

	Value	% of total
Avg. session length	N/A	N/A
Control packet O/H	1.12 kb/s	0.18 %
Avg. # of active multicast channels	1	N/A
O/H caused by multicast channels	0 kb/s	0 %
Effective data throughput	610.1 kb/s	99.82 %
Total throughput (data and control)	611.22 kb/s	100 %

data rate of 2 Mb/s. This result is due to the multihop forwarding restrictions on a common channel along the path $<A$-C-E-$F>$. When the source node A sends the packet, node C receives the packet and forwards it to node E. This initial forwarding process reduces the throughput to half of the original. One half of the channel is used for receiving the packet at node C and the other half for sending the packet to node E. When node E forwards the data packet to node F, the available channel bandwidth is further reduced to a third of the original. Namely, in optimal conditions the channel operates as a TDM channel with three slots per frame. Only one slot is active in any frame (in correspondence with the active hop). This performance degradation was already observed in the early packet radios unicast experiments [7]. Note also that we are using UDP because of multicasting.

Since the forwarding nodes relay the packets only if the forwarding group flag is set, no unnecessary forwarding exists. The dynamic adaptation scheme in ODMRP uses control packets to adapt to route changes caused by node movements or by changes in intermediate link quality. When there is a severe change in the link condition, DVMRP simply discontinues the forwarding of the multicast streams. Typical DVMRP route table update frequency is inadequate to track rapid topology changes of wireless networks. Because of this limitation, DVMRP can be used only in a relatively stable environment. Our experiments are based on a stationary network. Thus, as expected, ODMRP and DVMRP give comparable performances. If mobility were introduced into our testbed, link and topology changes would make ODMRP performance superior to DVMRP. Although we aimed the current stationary experiments at verifying the correctness of the ODMRP, future experiments will evaluate its efficiency in mobile scenarios.

V. CONCLUSION

We presented our experimental analysis of ODMRP (On-Demand Multicast Routing Protocol) for an ad hoc wireless network. ODMRP is based on mesh (instead of tree) forwarding. It applies on-demand, as opposed to periodic, multicast route construction and membership maintenance. We implemented ODMRP and performed the experiments on a six node wireless ad hoc testbed with Linux operating system. Our experiments consisted of forwarding MBone traffic from wired to wireless network using DVMRP gateway and routers, and streaming multicast traffic in the wireless testbed using ODMRP and DVMRP routers. We analyzed and tabulated the traffic data for experiments. Our experiments confirmed the fact that DVMRP with MBone multicast feed introduces high channel overhead because of the forwarding of unnecessary data stream caused by the control message losses in wireless channels. When replacing the MBone feed with a local source, DVMRP multicast overhead disappeared since there is only one source. In the local source multicast scenario, ODMRP and DVMRP performance is almost identical since control packet overhead is very low and comparable. DVMRP however, is not expected to perform well in mobile networks due to lack of inherent fast adaptivity. Therefore, ODMRP is expected to outperform DVMRP in a mobile environment. Our on going work includes deploying the network in an outdoor environment, introducing mobile nodes, increasing the network size, and building a hybrid router that converts DVMRP MBone feed into ODMRP-ready multicast.

References

[1] S. H. Bae, S.-J. Lee, W. Su, and M. Gerla, "The Design, Implementation, and Performance Evaluation of the On-Demand Multicast Routing Protocol in Multihop Wireless Networks," *IEEE Network*, special issue on Multicasting Empowering the Next Generation Internet, vol. 14, no. 1, January/February 2000, pp. 70-77.

[2] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees (CBT) - An Architecture for Scalable Inter-Domain Multicast Routing," *Proceedings of the ACM Conference on Communications, Architectures, Protocols and Applications (SIGCOMM'93)*, San Francisco, CA, October 1993, pp. 85-95.

[3] S. E. Deering and D. R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, vol. 8, no. 2, May 1990, pp. 85-110.

[4] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei, "The PIM Architecture for Wide-Area Multicast Routing," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, April 1996, pp. 153-162.

[5] H. Eriksson, "MBONE: The Multicast Backbone," *Communications of the ACM*, vol. 37, no. 8, August 1994, pp. 54-60.

[6] W. Fenner, "Internet Group Management Protocol, Version 2," *Request For Comments 2236*, Internet Engineering Task Force, November 1997.

[7] J. Jubin and J. D. Tornow, "The DARPA Packet Radio Network Protocols," *Proceedings of the IEEE*, vol. 75, no. 1, January 1987, pp. 21-32.

[8] S.-J. Lee, W. Su, and M. Gerla, "On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks," *Internet Draft*, `draft-ietf-manet-odmrp-02.txt`, January 2000, Work in progress.

[9] S.-J. Lee, W. Su, and M. Gerla, "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks," *ACM/Baltzer Mobile Networks and Applications*, special issue on Multipoint Communications in Wireless Mobile Networks, to appear, 2001.

[10] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols," *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 2000)*, Tel Aviv, Israel, March 2000, pp. 565-574.

[11] J. Moy, "Multicast Routing Extensions for OSPF," *Communications of the ACM*, vol. 37, no. 8, August 1994, pp. 61-66, 114.

[12] Multicast and MBONE on Linux - Application, downloadable from `http://www.teksouth.com/linux/multicast/applications.html`.

[13] W. van der Moolen, "IEEE 802.11 WaveLAN PC Card User's Guide," Lucent Technologies, June 1998.