VLM^2: A Very Lightweight Mobile Multicast System For Wireless Sensor Networks

Anmol Sheth, Brian Shucker, and Richard Han
University of Colorado, Department of Computer Science
Campus Box 430, Boulder, CO 80309-0430
{sheth, shucker, rhan}@cs.colorado.edu

***Abstract*—Wireless sensor networks require lightweight routing tailored for sensor devices with severe memory, power, and cost constraints. Such lightweight protocols must also support mobility and fault tolerance. The Very Lightweight Mobile Multicast (VLM^2) system addresses these concerns, introducing multicast support into wireless sensor networks. In simulation and in a true implementation on hardware Motes, VLM^2 achieves multicast with a lightweight footprint of no more than 17 Kb per node and also responds with agility to a wide range of mobility.**

***Index terms*—sensor networks, multicast, lightweight, routing**

I. INTRODUCTION

Wireless sensor networks (WSNs) introduce the need for efficient automatic routing among collections of resource-constrained wireless sensor nodes. The Very Lightweight Mobile Multicast system (VLM^2) is designed to support efficient lightweight multicast to mobile nodes in an evolving sensor network. A sensor network can consist of many intelligent sensor nodes deployed into an area to record and relay data about the environment. The sensor nodes form a multi-hop routing fabric that relays data to/from sensors. For such sensor networks, efficient multicast is a desirable network service, as the user is often interested in the attributes of a collection of nodes rather than in the individual nodes, e.g. "turn on all the sensors who monitor temperature and are located in a given area" [Intanagonwiwat]. Multicasting such a message offers greater energy efficiency, lower overhead, and more flexibility than unicasting the same message "turn on your temperature sensor" to each temperature sensor.

Sensor networks are often severely resource-constrained. Intelligent sensor nodes typically have very limited battery life, small form factor, and cost constraints that limit the memory, CPU speed, and complexity of the radio interface. Due to such resource constraints, sensor network protocols should be as lightweight as possible.

WSNs also introduce the challenge of mobility by sensor nodes. Sensors will be added to the network, change locations within the network, and leave the network (deliberately or because of failure). These events may be frequent in some applications. Thus, a sensor network protocol must tolerate node mobility and faults.

Figure 1 illustrates a typical WSN topology. Each node communicates with its neighbors within a limited range. VLM^2 assumes a hierarchical model in which there is at least one "basestation" of greater capability than the resource-constrained sensor nodes. In many sensor deployment scenarios, the data collected by the sensor network must be relayed back to a resource-rich wired infrastructure for further processing. A basestation node provides the bridge into the wired network, e.g. a wired/wireless gateway or satellite uplink. In Figure 1, the WSN forms a routing tree rooted at the basestation.

VLM^2 builds lightweight multicast support into resource-constrained WSNs that are characterized by hierarchy and mobility. Rather than building multicast on top of an underlying unicast network, VLM^2 is implemented directly on top of the link layer. This approach significantly reduces router state and code size. VLM^2 also adapts to mobility-induced changes in network topology resulting from nodes joining/leaving the network, or from nodes changing their location. Figure 1 shows how mobility by temperature node T2 changes the multicast routing tree. VLM^2 provides an integrated routing system capable of base-to-node multicast, broadcast, and unicast routing, as well as node-to-base unicast and broadcast routing.

Traditional multicast routing protocols for static wired networks are tree based protocols [Partridge, Moy, Ballardie, Deering], maintaining the tree via a global routing data structure like link state or distance vector. These structures are difficult to maintain in dynamic mobile environments. To address mobility concerns, various multicast routing protocols have been developed for wireless ad hoc networks [Lee, Liu, Lynn]. These protocols are often too heavyweight for resource-constrained WSNs.

Lightweight routing protocols for resource-constrained WSN's are beginning to emerge [Akyildiz]. BLESS is a BeaconLESS ad hoc routing protocol included in the standard TinyOS release for the Berkeley-designed sensor Motes [BLESS]. BLESS enables upstream unicast routing from sensor nodes to a basestation, but resorts to pure flooding downstream, and does not address downstream multicast. Lightweight Adaptive Multicast Protocol [Ji] provides multicasting service for large scale mobile ad hoc networks. LAM's tight coupling with the ad hoc Temporally-Ordered Routing Algorithm (TORA) [Park] creates a combination that is not so lightweight. AMRIS is a multicast protocol developed for hierarchical and mobile ad hoc wireless networks [Wu]. AMRIS is relatively lightweight in terms of state, but not bandwidth, since control

messages are sent periodically rather than being data-driven. VLM^2 efficiently leverages the flow of data packets to update the routing tree, and only resorts to control overhead in the absence of data.

II. PROTOCOL SPECIFICATION

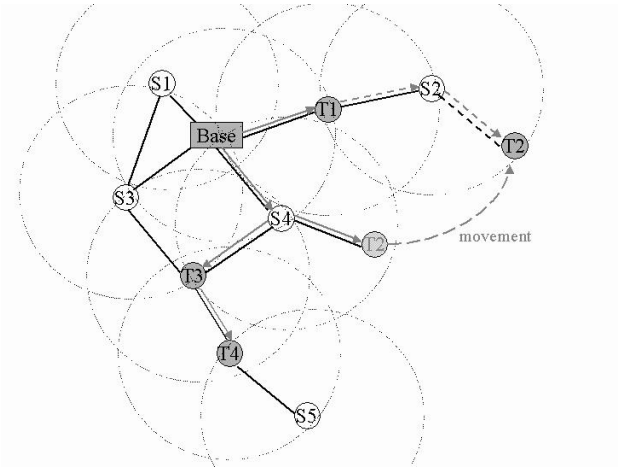

Figure 1. A wireless sensor network with a root basestation multicasting to temperature sensors T1 – T4. Motion by node T2 changes the topology of the multicast tree.

VLM^2 provides three forms of unreliable datagram service in the base-to-node downstream direction, namely multicast, unicast and broadcast, and both unicast and broadcast packet service in the node-to-base upstream direction. Intelligent nodes may subscribe to any number of multicast groups. A multicast group may contain any subset of the nodes. Downstream unicast is accomplished by declaring a multicast group with only one member.

VLM^2 supports WSNs with arbitrary topology, so long as there is at least one basestation. In the current implementation, the basestation is predetermined; election of the basestation is possible but beyond this paper's scope. Once a basestation has been identified, other nodes may join the network. VLM^2 will essentially build a spanning tree rooted at the basestation.

Individual nodes contain only a modest amount of local routing state: a list of identifiers of its downstream multicast groups; and a cache of packet headers to avoid retransmission of previously seen packets. This downstream groups list is currently maintained as a bitmask to reduce storage.

VLM^2 uses 8-bit identifiers for nodes and for multicast groups. This restricts the network to 255 nodes and 248 multicast groups. The basestation has a reserved multicast identifier, and the flooding address has one as well. The small address space reduces packet overhead; increasing the address space to 16 bits would be trivial.

Figure 2 shows a VLM^2 packet containing the following relevant header fields:

- source field contains the identifier of the node that originated the packet.
- immediateSource field contains the identifier of the node that last forwarded the packet.
- seqno contains a sequence number that is set by the node that originates the packet. This sequence number is incremented with every packet, to make each packet's (source, seqno) pair unique.
- dest contains the multicast group identifier. Packets coming from the basestation may have any destination, but packets coming from a regular node may only be sent to the basestation or flooded.
- route is used only for upstream packets and indicates the next upstream node that should forward the packet. It is updated by every node that forwards the packet.
- TTL contains the current time-to-live in hops of this packet. The packet's TTL is set to a fixed initial value (currently 20), which is a constant across the entire network to eliminate routing loops.
- DtB field, which stands for Distance to Base, is the number of hops from the originator of the packet to the basestation.

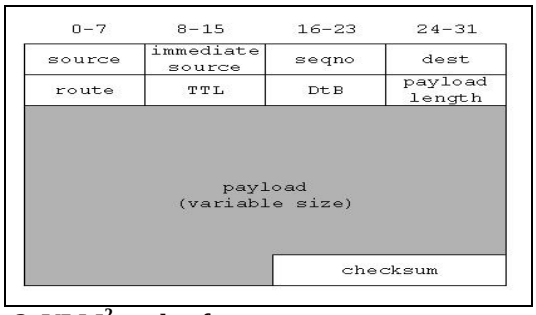

Figure 2. VLM^2 packet format

A. Broadcast

When a node receives a packet destined for the broadcast address, it inspects the packet before deciding to retransmit it. To reduce loops, each node maintains a small FIFO cache of (source, seqno) pairs to identify recently received packets. If a packet is in the cache, it is ignored. Every node in the network will transmit a flooded packet once and receive it once for every node it has a direct link to. In practice, a cache as small as 10 entries effectively eliminates loops, but the TTL is what guarantees that loops cannot persist.

B. Name Assignment

When a node starts up, it must send a control packet, NAME_REQUEST, to the basestation to obtain an address or identifier. The nodes closest to the basestation receive their id's first, and the network builds its routing fabric of known nodes from the inside out. When the basestation receives the NAME_REQUEST, it chooses an identifier for the new node and responds by sending a NAME_ASSIGN packet indicating the new ID, currently flooded, though we plan to implement a

more efficient reverse-path solution. If an assignment is not received after a given timeout period, the node repeats the procedure.

To distinguish multiple nodes making name requests at the same time, the request packet includes a unique large identifier, such as a hardware address or a random number. This large ID is echoed back in the NAME_ASSIGN packet, so the node can identify the correct response.

C. Distance to Base

Each node maintains a Distance to Base (DtB), which is the number of hops from the node to the basestation. At startup, a node's DtB equals the network's uniform initial TTL. When a node receives any packet from the basestation, the packet's TTL field is checked to determine how many hops have been traversed. If the number of hops is lower than the current DtB, then the node must be closer to the basestation than before, so its DtB is updated to the new value. If the number of hops is greater than the current DtB, then either the node has moved away, another node on the path to the basestation failed or moved away, or the packet did not take the shortest route to the node.

A timeout mechanism determines when to increase the DtB. Each node maintains a timer indicating the last time it updated its DtB. When a packet from the basestation indicates that the DtB is correct or should be lowered, the timer is reset. When the timer expires, the node increments its DtB by one and sends out a DTB_UPDATE packet. This packet is never forwarded; only the immediate neighbors respond. Upon receipt of a DTB_UPDATE, if the indicated new DtB is equal to or greater than the neighbor's own DtB plus one, then the neighbor sends a DTB_REPLY packet to the updating node informing the updating node that the neighbor has a route back to the basestation that is the same or better than the updating node's current path.

The updating node will continue to increment its DtB and send out DTB_UPDATE packets until either it receives a DTB_REPLY or its DtB reaches the network's initial TTL value. The latter case will occur if the basestation is unreachable because of a network partition, and will leave the updating node in a state resembling startup.

D. Node-to-Base Unicast Routing

When a packet destined for the basestation is received at an intermediate node, the sender's distance to the basestation is known from the DtB field, and the number of hops taken already is found by subtracting the TTL field from the known initial TTL. The intermediate node also knows its own distance to the basestation. If (the sender's DTB) - (hops already taken) >= (intermediate node DTB), then the node knows it is on the shortest path from the sending node to the basestation, and forwards the packet; otherwise the packet is dropped. At this point, packets would take **all** of the shortest paths from the sending node to the basestation. To restrict packets to a single path, each node keeps track of one node directly upstream from it. When sending or forwarding a packet upstream, the identity of the upstream node is placed in the packet's route field. Only the node indicated in the route field will forward the packet, creating one unique shortest path.

The upstream node is the immediate sender that caused the DtB timer to be reset. The DtB timer is reset upon receiving a packet indicating a DtB that is equal to or less than the current DtB value. The "equal to" case allows packets to be sent through the upstream node that was most recently heard from. This ensures that the node will be quickly reconnected if it moves or a node upstream from it fails.

E. Base-to-Node Multicast Routing

To build a multicast routing tree, nodes may subscribe to an address by sending a SUBSCRIBE packet containing a group identifier. SUBSCRIBE packets get routed to the basestation as described in Section E. Every node maintains a list of group identifiers called a "downstream groups list." When a SUBSCRIBE packet is received at an intermediate node, it uses the caching algorithm to determine if it should forward the packet. If it does forward the packet, then it also adds the associated group identifier to its downstream groups list, indicating that some node downstream is a member of that group. Whenever a packet addressed to a group is received at a node, the packet is forwarded if the packet's destination address in the node's downstream groups list. Duplicate packets already in the cache are never forwarded.

Entries in the downstream groups list expire, so nodes must periodically re-subscribe to the groups. The intermediate nodes contain only soft state—if one fails, the multicast connection will be reestablished by the next SUBSCRIBE message. Also, when a group member leaves the network, its memberships will eventually just expire. As an optimization, re-subscription may be triggered by certain events, such as DtB updates.

III. PERFORMANCE ANALYSIS

Network simulation of VLM^2 was used to verify the protocol's correctness and to analyze its responsiveness to mobility and its overhead cost. The network consisted of ten nodes and a basestation. At fixed intervals, each node sends a data packet to the basestation. At startup, each node joins one multicast group in which it is the only member. The basestation periodically sends a data packet to each multicast group.

To introduce routing errors similar to those that would result from node mobility, the simulator periodically chooses two nodes at random and swaps their positions in the network. Swapping changes the routing but not the topology. Packet loss and control overhead in a VLM^2 network were measured as a function of node mobility. Node mobility is measured via

two ratios: the DtB timeout period divided by the period between node swaps, and the multicast subscription expiration period divided by the same swap period.

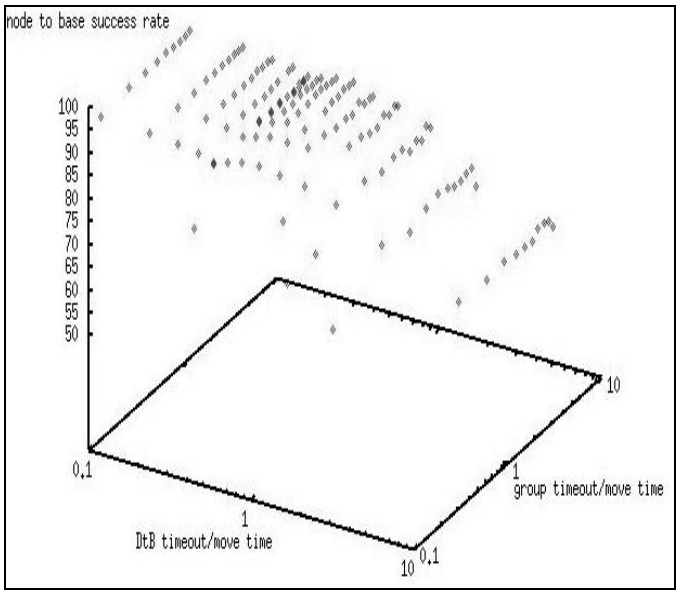

Figure 3. Upstream efficiency

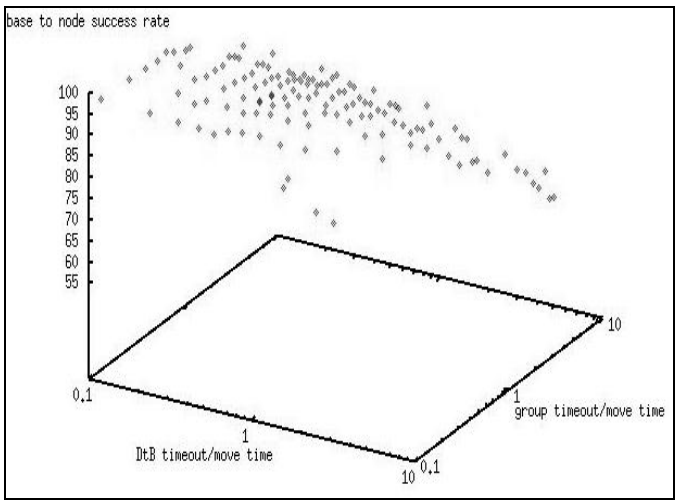

Figure 4. Downstream efficiency

As shown in Figures 3 and 4, routing from the nodes to the basestation is effective when the DtB timeout is smaller than the swap period. In Figures 3 and 4, increased mobility (large DtB timeout/move time > 1) causes heavy packet loss in both node-to-base and base-to-node communication, since the network is not reacting fast enough to changes in the topology, causing lost packets. In Figure 4, we also see that slower multicast subscriptions relative to the swap time (large group timeout/move time ratio) increased the packet loss, since base-to-node communicatoin is dependent on packets travelling both upstream (subscriptions) and downstream (data). If a node or its neighbors are highly mobile, it must refresh its subscriptions more often to avoid losing messages to topology changes.

In figure 5, control overhead is defined as the percentage of packets that do not contain data from the application layer. Control messages in VLM^2 are driven both by timeouts and by events. Initially, overhead decreases as expected with longer timeouts, as the frequency of control messages decreases. However, at some threshold, control overhead increases as the nodes become more mobile, because the frequency of error conditions requiring control messages to repair the network will also increase. There is no advantage to setting timeouts longer than this threshold; on the other hand, configuring the network with progressively shorter timeouts trades off overhead for reliability.

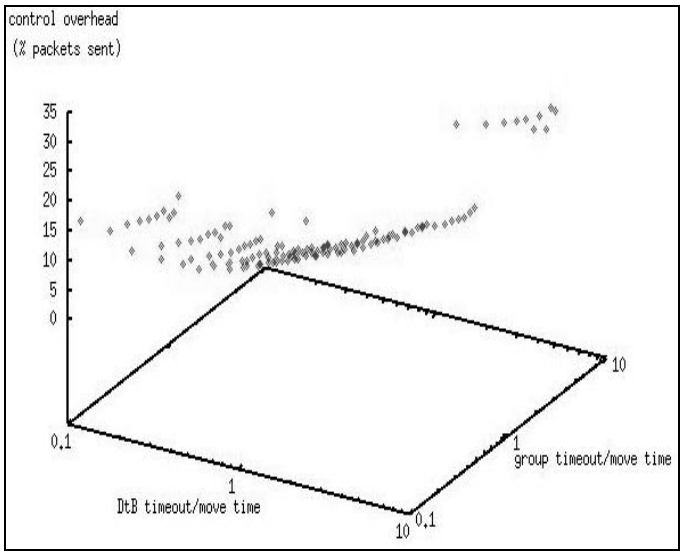

Figure 5. Overhead vs Mobility

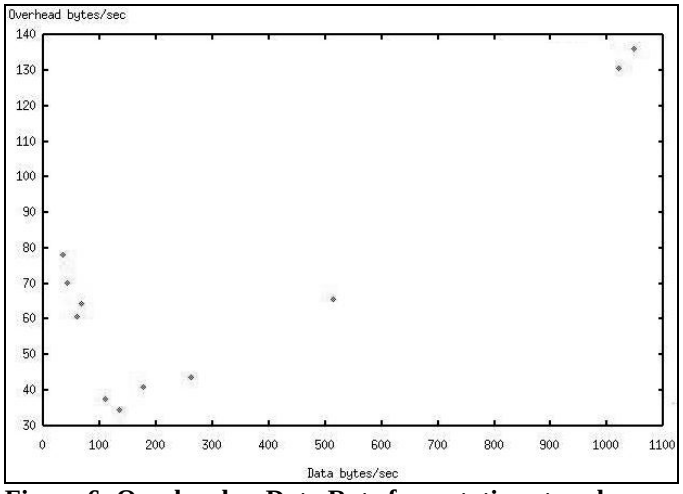

Figure 6: Overhead vs Data Rate for a static network.

Figure 6 shows control overhead (in bytes transmitted, not packets as in the previous figure) as a function of the data rate in a static network. Unsurprisingly, there is some overhead associated with sending data, which explains the near-linear upward trend as data rate increases. When the data rate is very low, additional control messages--specifically DtB updates and replies--are required to maintain the routing infrastructure. As

data rate increases, DtB information is inferred from the data packets, and this overhead dissappears.

IV. HARDWARE IMPLEMENTATION

VLM^2 was implemented over a collection of MICA sensor motes, as shown in Figure 6. The resource constraints were:

- a 4MHz processor with 128 Kb of program memory and 4 Kb of data memory
- an RFM Monolithics TR1000 radio at 19.2kbps
- 2 AA batteries for power

The motes are pre-installed with the TinyOS Tiny Microthreading Operating System [Hill], a small, open source, energy efficient operating system developed for sensor networks. The VLM^2 protocol was implemented using the modified lightweight Active Message (AM) paradigm [vonEicken] for message based communication in TinyOS. The implementation of VLM^2 can be logically partitioned into three main components:

- Processing of packets on a basestation
- Processing of packets on multicast tree nodes
- Interface between the basestation and PC.

VLM^2 is very lightweight in terms of consuming very little memory for both data state and code size. The static state information required by VLM^2 for each sensor node in the multicast tree occupies only a total of 65 bytes of data. Of these 65 bytes, the bulk is consumed by 20 bytes of message cache (10 entries of 2 bytes each) and one AM data buffer of 36 bytes. In addition, the code size of VLM^2 is quite modest. The total size of the VLM^2 binary file that's transferred onto the mote, including the size of the TinyOS, is only 17 KB. This confirms our original claim that VLM^2 leaves a very lightweight memory footprint suitable for deployment in WSNs. VLM^2 is also lightweight in terms of consuming little bandwidth. The data-driven nature of the protocol reduced the amount of control packet overhead. Control messages such as the DTB_UPDATE only appeared after a timeout of one second, triggered by the absence of data.

VLM^2 displayed great agility in adapting to motion. We designed an experiment in which each node would blink a green LED if it was a leaf node on the multicast tree, or yellow otherwise. Nodes were rearranged by hand to induce mobility. The multicast tree reconfigured itself quickly. Typically, new leaf nodes were identified within 3 seconds, while old leaf nodes turned into intermediate nodes due to mobility were also identified within 3 seconds, despite the limitations of the 19.2 kbps link and a one second timeout.

V. CONCLUSION

This paper presents a very lightweight mobile multicast system VLM^2 for wireless sensor networks. The VLM^2 system is distinguished by its lightweight footprint, data-driven design, adaptation to mobility, and integrated support for multicast, broadcast and unicast. The protocol was first tested via simulation and then implemented on mote-based platforms to form a true mobile multicast-based wireless sensor network.

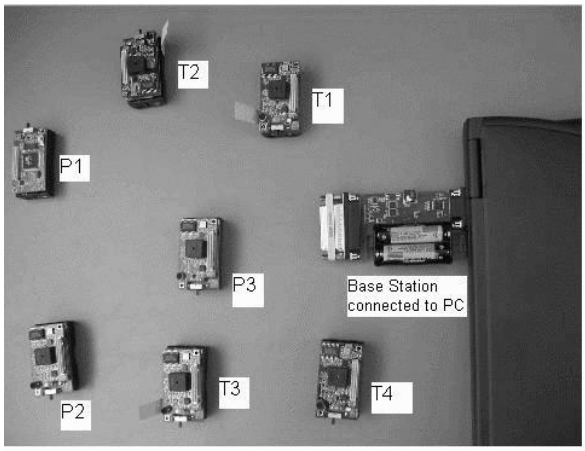

Figure 6. Very Lightweight Mobile Multicast (VLM^2) system builds a multicast tree rooted in the basestation over wireless sensor Motes.

VI. REFERENCES

[Akyildiz] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. *A Survey on Sensor Networks*. IEEE Communications Magazine, August 2002. Page(s) 102-114.

[Ballardie] A.J. Ballardie, P.F. Francis and J. Crowcroft. *Core Based Trees*. Proceedings of ACM SIGCOMM, San Francisco, 1993. Page(s) 85-95.

[BLESS] http://webs.cs.berkeley.edu/tos/

[Deering] S, Deering, D. Estrin, et al. Protocol Independent Multicast – Sparse Mode (PIM-SM): Motivation and Architecture, draft-ietf-idmr-pim-arch-04ps, October 1996.

[Hill] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. "System architecture directions for network sensors". In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA, USA, Nov. 2000. ACM. Page(s) 93-104.

[Intanagonwiwat] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," MobiCom 2000, pp. 56-67.

[Ji] L. Ji, M.S.Corson, A lightweight adaptive multicast algorithm, Proceedings of the IEEE Globecom 1998, Page(s): 1036 -1042 vol.2

[Lee] S. Lee, W. Su and M. Gerla "On-Demand Multicast Routing Protocol", Proceedings of WCNC, September 1999 Page(s) 1298-1302

[Liu] M. Liu, R.Talpade, A. McAuley, E. Bommaiah, "AMRoute:Adhoc Multicast Routing Protocol", Technical Report 99-1, The Institute for Systems Research, University of Maryland, 1999

[Lynn] G.H.Lynn,T.F. Znati "RoMR: A Robust multicast routing protocol for ad-hoc networks" Local Computer

Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on, 2001 Page(s) 260-268.
[Moy] J. Moy, "*MOSPF: Analysis and experience*," RFC 1585, Network Working Group, Mar. 1994.
[Park] V.D. Park and M. Scott Corson. “A highly adaptive distributed routing algorithm for mobile wireless networks.” In Proceedings of lNFOCOM April '97, pages 1405-1413,
[Partridge] C. Partridge D. Waitzman and S. Deering. RFC 1075, Distance Vector Multicast Routing Protocol. SRI Network Information Center, November 1988.
[vonEicken] T. von Eicken, D.E. Culler, S.C. Goldstein, and K.E. Schauser. “Active Messages: a Mechanism for Integrated Communication and Computation.” In The 19th Annual International Symposium on Computer Architecture, Gold Coast, Australia, May 1992. Page(s) 256--266,
[Wu] C.W. Wu and Y.C. Tay, "AMRIS: A Multicast Protocol for Ad hoc Wireless Networks," In Proceedings of IEEE MILCOM'99, Atlantic City, NJ, Nov. 1999. Page(s) 25-29