Implementation and Validation of Multicast-Enabled Landmark Ad-hoc Routing (M-LANMAR) Protocol

Yunjung Yi, Joon-Sang Park, Sungwook Lee, Yeng-Zhong Lee, and Mario Gerla
Wireless Adaptive Mobility Lab
Computer Science Department
University of California, Los Angeles, CA 90095
Email: {yjyi, jspark, swlee, yenglee, gerla}@cs.ucla.edu

ABSTRACT

In this paper, we investigate the performance of Multicast-Enabled Landmark Ad-hoc Routing (denoted as M-LANMAR) protocol using a Linux implementation as well as a QualNet simulation model. The main objectives are the validation of our implementation and the demonstration of the advantages of M-LANMAR. Using a Linux implementation of ODMRP, we compared the performance of M-LANMAR to that of ODMRP in a small-scale testbed. Through simulation, we show also the scalability of M-LANMAR to large networks.

1. INTRODUCTION

Recent advances in wireless ad hoc communications, robotics and microflyer technology will enable the deployment of large-scale network with autonomous agents such as small unmanned ground vehicles (UGVs), sea and airborne vehicles. Those agents can be clustered i.e., grouped as a team based on their characteristics. In particular, nodes in the same team will have coordinated motion. We call this model the “affinity team model”. For example, a team consists of UGVs within a certain area or unmanned airborne vehicles (UAVs). And it is possible to launch complex missions that comprise several such teams. Possible applications include: coordinated aerial sweep of vast urban/suburban areas to track suspects, search and rescue operations in areas unfriendly to humans (e.g., chemical spills, terrorist attacks, etc), exploration of remote planets, reconnaissance of enemy field in the battle-theater, etc.

The affinity team model considerably simplifies the mobility management problem and allows us to design a routing protocol that scales. In fact, it suffices for a source to know the path to one of nodes in the team (say, a landmark) in order to route a packet to any other destination within that team.

We developed an efficient and scalable multicast protocol called Multicast-Enabled Landmark Ad hoc Routing Protocol (M-LANMAR) [1]. Our approach exploits the motion affinity (more precisely, coordinated motion [2]) that exists among the nodes in the same team. Each team can be viewed as a logical subnet. Within the team a representative node (say a “landmark”) is dynamically elected. The address of and the path to the landmark are propagated (advertised) into the entire network. Thus, every potential source knows how to reach any team. Moreover, if the landmark also advertises the multicast groups (i.e., missions) to which the team belongs, the source need not know which members are parts of the multicast group. It simply inspects the landmark table (dynamically updated using the landmark advertise/). and checks the multicast group fields. M-LANMAR creates a tunnel from the multicast source to each landmark of the subscribed teams in the multicast group. It then sends a separate copy of the packet to each landmark (i.e., multiple unicast). Once a landmark sees a packet, it initiates scoped flooding to deliver the packet to the entire team members.

In [1], we showed that M-LANMAR works effectively compared to a traditional “flat” ad hoc multicasting protocol such as On-Demand Multicasting Protocol (ODMRP) [3] and flooding through extensive simulation study. In this paper, we further present our real implementation of M-LANMAR developed on Linux platform and its validation using a small scale network scenario. Due to the scale limitation of real experiments, we show the scalability of M-LANMAR through simulation study.

The rest of our paper is organized as follows. In Section 2, we describe the protocol of M-LANMAR. And Section 3 introduces the implementation issues. We show our experimental result in Section 4 and simulation results in Section5. Finally we conclude our paper in Section 6.
Multicast-enabled Landmark Ad Hoc Routing (M-LANMAR) protocol extends Landmark Ad Hoc Routing (LANMAR) [1] protocol. M-LANMAR is divergent from existing mobile ad hoc network (MANET) multicast protocols such as ODMRP (On-Demand Multicast Routing Protocol [3], MAODV [4], CAMP[5]) in that M-LANMAR aggregates unicast routing table update and multicast routing management. Thus, M-LANMAR achieves constantly low route maintenance cost regardless of dynamic membership changes (e.g., the increasing number of members and multicast groups). Furthermore, M-LANMAR, unlike traditional general purpose MANET multicast protocols, maximally exploits the group affinity model by extending LANMAR [6] that works effectively with affinity team model.

M-LANMAR protocol is a proactive scheme, where group membership and multicast routes are updated proactively. With the aid of an underlying unicast protocol, the sources maintain the multicast routes to only landmarks of joined teams instead of individual paths to each member.

A. LANMAR

In LANMAR, each Landmark advertises its presence in the network and the path to reach it with periodic update packets. The address <team ID, Host ID>, compatible with IPv4 and IPv6 formats, will direct a packet towards the landmark of the team, following the advertised path. A "local", limited scope routing scheme is used to route packets within a few hops from the source. Given the very limited scope, local routing can be supported by a MANET proactive scheme such as DSDV (Destination Sequenced Distance Vector), Fisheye Routing [6] or OLSR (Optimal Link State Routing)[7]. Packets directed to another team are routed to the corresponding Landmark first, following the advertised path. Once in the proximity of the destination, the local scope routing takes over. It is assumed that nodes in the same team are reachable within at most two or three radio hops from the landmark. Thus, the majority of intra team communications within a team and some of the communication across closely cooperating teams simply use local routes. A snapshot of the LANMAR operation is shown in Figure 1.

B. Join Multicast Group

In LANMAR, each node keeps fresh routes to all landmarks in the network by periodic landmark updates. Using the landmark updates, a team maintains its membership to multicast group(s). A landmark of a team that wishes to join the multicast group(s) implicitly advertises "Join Request" to the sources by piggybacking the targeting multicast group ID(s) (address(es)) on landmark broadcast packet. Upon receiving the "implied" Join Request, each node in the network updates respective landmark entry with the subscribed multicast group IDs. Thus, the Join Request will be propagated into the sources in a few landmark table exchanges. Membership is constantly refreshed, as each landmark includes subscribed multicast addresses to all outgoing landmark update packets.

C. Leave Multicast Group

When a team who is a part of multicast group wants to leave, the landmark removes the ID of that multicast group from its subscribed multicast groups list. Thus, the
landmark will stop advertising the group. The landmark's entry at other nodes will be updated accordingly.

D. Data Propagation

The source nodes look up their landmark table to find the landmark addresses of the subscribed teams. For each landmark that subscribes to this multicast group, the source creates a "virtual link", i.e., a tunnel, to the landmark and sends encapsulated multicast data. Upon reception of the encapsulated data, each landmark initiates flooding within the subnet so that each member can receive the data (see Figure 2). With an assumption of restricted size of the subnet ("x" hops from the landmark to all nodes), we use local flooding with initial TTL "x+1" (in our simulation x =2). Each node in the team accepts incoming multicast data.

At first glance, the "multiple unicast" approach may seem inefficient. In fact, one may reduce the link overhead by using conventional ad hoc multicast (e.g., ODMRP) to the landmarks. Moreover, the scalability issue has now been resolved by the mere use of landmarks. However, there are still problems remaining in conventional multicast. First, the most popular multicast schemes, i.e., ODMRP and MAODV are "on demand" schemes. Whenever the source wishes to send a multicast message, it must first set up the multicast tree or mesh. This introduces latency (up to seconds) that may be unacceptable in the real time coordination and control of a mission (for example, multiple sensor beamforming). M-LANMAR in contrast proactively maintains the paths to the Landmarks all the time.

A second benefit of tunneling is reliable data delivery. It is well known that multicast (as opposed to unicast) in an ad hoc network is unreliable and prone to loss for two key reasons: the multicast MAC layer is NOT protected against hidden terminals, and; TCP cannot be used on top of multicast (because of ACK implosion). Thus, only a fraction of the teams receives the multicast packet. In some applications, a small loss is tolerable (for example, video streaming). However, other applications (e.g., mission level coordination of the teams) require that ALL teams receive the packet correctly. Else, mission synchronization may be lost. M-LANMAR achieves this goal by simply using a robust, MAC layer unicast, and by running TCP on the tunnel from source to Landmark. The final distribution of the multicast packet within the team is very reliable as it uses local scoped "flooding".

The last, but not least benefit of M-LANMAR is the protection against congestion. Congestion is always a major concern in ad hoc network, especially networks that carry time critical data. Open loop traditional multicast is NOT protected against congestion. Some proposals (for reliable, congestion controlled ODMRP and MAODV, for example) have been published, but have not been proven to be completely robust. M-LANMAR congestion can be controlled in various different ways. One way is to use TCP. The TCP congestion control window automatically guarantees congestion protection. Moreover, TCP provides rapid feedback to the source about the congestion in one or more teams. Then, the source can enforce precedence policies and for instance, transmit only control traffic (and postpone low priority data traffic) while congestion persists. This is not possible in multicast schemes as they are not equipped with feedback.

III. IMPLEMENTATION

We implemented M-LANMAR as a user-level daemon running on the Linux. Thus, our implementation did not require any modification of Linux kernel. The implementation was developed on the Linux kernel version 2.4.19 that came with Mandrake Distribution 9.0. All necessary software pieces including wireless network interface drivers and compilers were also provided by the distribution.

Two components constitute the M-LANMAR implementation: routing daemon and packet forwarding engine. The routing daemon was first designed to implement LANMAR unicast routing protocol and then extended to support M-LANMAR functionality. It assumes that the subnets (teams) are predefined and assigned unique IDs. In each subnet, one landmark is elected using the lowest ID wins [10] rule (a clustering algorithm). The routing daemon in a system periodically exchanges (say every second) two kinds of distance vectors with peer nodes: (1) Landmark distance vector information (LMDV) to update landmark routing table; and (2) local distance vector to maintain and update the local routing table. Multicast group membership information is piggybacked on LMDV messages. Each landmark entry includes the list of its joined multicast groups. Our routing daemon, designed to replace the default routing daemon routed, updates kernel routing tables according to its two routing tables. By setting the kernel route table properly we can expect the kernel to forward unicast traffic. While we use Linux kernel's unicast traffic forwarding service, we do not rely on Linux kernel's multicast forwarding capability, but provide our own packet forwarding engine (PFE) tightly coupled with the routing daemon as M-LANMAR defines a new multicast packet delivery scheme.
In M-LANMAR, multicast traffic is unicast-tunneled from a source to landmarks first and then the scoped flooding for intra-team delivery is initiated at each landmark. At the sending node, PFE intercepts every outgoing IP multicast packet, and encapsulate it within a UDP or TCP unicast packet. On receiving the unicasted packet at the landmarks, PFE broadcast the packet for scoped flooding. PFE on non-landmark nodes rebroadcast every incoming broadcasted packet after checking its TTL for scoping and ID for duplicate packet suppression. When rebroadcasting, jitter, i.e. some random wait, is introduced to reduce conflicts between the rebroadcasts. Forwarding the multicast packets to the destination multicast application from PFE is done by loopbacking the packets. Our M-LANMAR implementation is compatible with any existing multicast applications using the standard socket interface. Intercepting outgoing packets and loopbacking incoming packets makes M-LANMAR transparent to any existing socket-based multicast applications.

IV. EXPERIMENTS AND RESULTS

A. Testbed Configuration

Our testbed consists of five Dell 1.8 GHz Pentium 4 Latitude C840 laptops equipped with Orinoco 802.11b PCMCIA card. The testbed is designed to verify that our implementation of M-LANMAR follows the protocol description as described in the previous section. The topology of the testbed is illustrated in Figure 3. There are one multicast source, node 1, one intermediate node, node 2, and three receivers, node 3, 4 and 5, constituting three subnets/teams represented by dashed circles. Receivers are two hops away from the sender. All nodes run on Mandrake Linux distribution 9 with kernel version 2.4.19. Linux PCMCIA package version 3.2.0 and Orinoco wavelan2_cs driver have been used for 802.11b devices and the devices are set to ad-hoc mode. All experiments are conducted in a radio channel locally reserved to minimize the interference. We developed a multicast/unicast traffic generator and collector to see how many packets are eventually delivered to a receiver.

B. Experiments and Result Analysis

As M-LANMAR aims to achieve scalability on large-scale networks, it is difficult to show full performance advantage with the experiments on a small-sized testbed. We rather focus on the verification of our implementation. We compare M-LANMAR to two reference models: simple unicasting and the Linux ODMRP implementation [8]. Since M-LANMAR uses unicasting in its two-level data dissemination scheme, it should show comparable results to that of pure unicasting. We define two metrics: (1) the packet delivery ratio -- the percentage of packets successfully reaching a receiver; and (2) the normalized control overhead -- the total number of control packets generated is divided by the total number of delivered data packets. We collect measurements under various traffic conditions using different payload sizes and packet transmission rates.
Multicast traffic streams are generated from node 1 towards node 3, 4 and 5. Unicast traffics are injected from node 1 to node 3. The results are summarized in Figure 4 and 5. For M-LANMAR and ODMRP, the delivery ratio is averaged over 3 receivers. We use the same one-second period both for the routing table update interval in M-LANMAR and the route refresh interval in ODMRP.

Figure 4 and 5 demonstrate the performance comparison of M-LANMAR and ODMRP over various traffic conditions. In the figures, we observe that (1) the delivery ratio of M-LANMAR is very close to that of unicasting because M-LANMAR transmits a data packet through unicast tunneling. This fact validates the first-level data dissemination from a source to each landmark of our M-LANMAR implementation; (2) M-LANMAR achieves higher delivery ratio near to 100% compared to ODMRP. This result shows that unicast transmission using RTS/CTS, acknowledgement and retransmission accomplishes the higher reliability than broadcast mechanism without RTS/CTS. Note that ODMRP uses MAC-layer broadcast; (3) Due to the flooding used for intra-team data delivery, all nodes in the same team see approximately same delivery ratio. To clearly show this, we introduce another metric, complete delivery ratio, defined as the number of packets received by all the receivers divided by the total number of packets sent. As shown in Figure 4, the delivery ratio and complete delivery ratio of M-LANMAR are very close to each other. Implicitly, this validates our implementation of intra-team flooding; (4) M-LANMAR shows higher control overhead than ODMRP. On-demand scheme employed in ODMRP is efficient with small number of active participants, i.e. senders and receivers, which is the case of our testbed. But on-demand protocols suffer from high control overhead when the population of senders and/or receivers increases in large-scale networks. The scalability of M-LANMAR with respect to control overhead is not evident in small-scale experiments. We will show the scalability through simulation in the following section.

V. M-LANMAR ON LARGE-SCALE NETWORKS

In the previous section, we verified the implementation of M-LANMAR. Thanks to unicast tunneling mechanism, M-LANMAR shows the higher delivery ratio than ODMRP even in the small-scale testbed. In this section, we show the advantages of M-LANMAR in large-scale networks through simulation.

![Figure 6: Delivery ratio in Mobile Network](image)

For this simulation study, we implemented M-LANMAR using QualNet [7] simulator. We use default parameters provided by QualNet. In our simulation, each source generates data in a CBR (Constant Bit Rate) fashion with UDP (User Data Protocol). We use IEEE 802.11 DCF MAC and two-ray ground path-loss model for the Channel. The transmission range of each node is 376m and bandwidth of the device is 2MBits/sec.

In the network, 1000 nodes are uniformly placed within 6000 x 6000 m² terrain and grouped into 36 teams. The average number of neighbors for each node is 10 and the average hop count from the landmark node to each node in the logical subnet is 2. For maintaining the routing structures, ODMRP uses 2 seconds interval for each Join Query and M-LANMAR uses 1 second interval for landmark updates and 2.3 seconds period for local routing table exchanges. In the scenario, node moves following the "Reference Point Group Mobility" model [9] with speed 2m/s with 10s pause time. In our simulation study, we use one source node and 3 teams for each multicast group. The source sends out four packets every second with 512 bytes packet size as default.

In Figure 6, we compare the performance of M-LANMAR with ODMRP and flooding scheme. This result demonstrates three important facts. First of all, it clearly shows that M-LANMAR works far better than ODMRP even in the presence of node mobility. Secondly, the performance of ODMRP degrades as we increase the number of groups. The reason of this performance degradation of ODMRP mainly comes from Join Query flooding for each multicast group.
ODMRP suffers from heavy contention and collision due to the increase of control overhead and the number of relayed packets. On the other hand, M-LANMAR shows stable throughput regardless of the number of multicast groups because it keeps the network stable using multicast group aggregation. All these observations put together indicate that M-LANMAR provides a scalable multicast solution. The analysis of flooding shows that the delivery ratio in flooding drops with heavy offered load as shown in Figure 6. We could not even complete the execution of the flooding runs with a large number of multicast groups (> 8) due to heavy memory requirements.

Figure 7 shows the normalized control overhead of ODMRP and M-LANMAR. This result demonstrates that the normalized control overhead of ODMRP slightly increases as the offered load becomes heavy (i.e., the number of multicast group increases). In fact, the total control overhead of ODMRP is proportional to the number of multicast groups whereas, in M-LANMAR, nodes exchange their local routing table and landmark table periodically regardless of actual offered load. Thus, the normalized control overhead of M-LANMAR decreases as the actual offered load increases.

VI. CONCLUSION

In the course of this discussion, we intended to share our experience on implementing and validating a scalable multicast protocol designed for large-scale mobile ad-hoc networks namely M-LANMAR. We have implemented M-LANMAR on Linux and performed experiments on a simple testbed to verify our implementation. The experimental results showed that by defining a new data delivery paradigm, M-LANMAR considerably improved data delivery compared to ODMRP. Via simulation, we also confirmed the scalability of M-LANMAR in large networks.

REFERENCES