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
Computer Science Department
University of California, Los Angeles, CA 90095
{yjyi, jspark, swlee, yenglee, gerla}@cs.ucla.edu
310-206-8589

Abstract- In this paper, we investigate the performance of M-LANMAR by implementing the protocol in Linux platform. With existing Linux implementation of ODMRP, we compare the performance of M-LANMAR to that of ODMRP. The two components constitute M-LANMAR implementation: routing and packet forwarding. Because M-LANMAR requires packet manipulations (e.g., the source duplicates the packet and each landmark initiates the scoped flooding), along with routing daemon, user-level packet forwarding engine for data manipulation is indispensable. Furthermore, we verify our implementation by analyzing the result compared to the simulation result.

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 advertises also 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 (which it dynamically updates using the landmark advertisements) and checks the multicast group fields. M-LANMAR creates a tunnel from the multicast source to each landmark of the subscribed multicast group (of teams). It then sends a separate copy of the packet to each landmark (i.e., multiple unicast). Once the packet has reached the target landmark, it is broadcast to all nodes within the team using restricted flooding. 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 the paper, we further implement M-LANMAR in Linux platform and show the validation of our implementation.

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 experiment result in Section 4 and finally conclude our paper in Section 5.

2. M-LANMAR

Multicast-enabled Landmark Ad Hoc Routing (denoted as M-LANMAR) protocol is divergent from existing MANET multicast protocols such as ODMRP (On-Demand Multicast Routing Protocol [3], MAODV [4],
M-LANMAR aggregates unicast routing table updates and multicast routing maintenance. Thus, M-LANMAR achieves constantly low maintenance cost, because of the underlying hierarchical routing LANMAR, 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.

2.1. 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.

2.2. 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.

2.3. 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 1). 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, unicast MAC layer, 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”.

![Figure 1: M-LANMAR Routing Protocol](image)
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.

3. Implementation

We implemented M-LANMAR as a user-level daemon running on Linux. No kernel modification was necessary. The implementation was developed on Linux kernel version 2.4.19 came with Mandrake Distribution 9.0. All necessary software pieces including wireless network interface drivers and compilers were also provided by the distribution. For the hardware platform, we used Dell Inspiron 8000 equipped with Lucent’s Orinoco 802.11 wireless LAN card.

The two components constitute M-LANMAR implementation: the routing daemon and packet forwarding engine. The routing daemon (RD in short hereafter) was first designed to implement LANMAR unicast routing protocol and then we extended it to support M-LANMAR functionality. It assumes that teams are predefined and assigned unique subnets. Dynamic landmark election based on lowest-ID is incorporated. LANMAR is a distance vector (DV) routing scheme. Two kinds of DV messages are periodically exchanged between immediate neighbors: Landmark DV (LMDV) message, and local node DV (LNDV). Multicast group membership information is piggybacked on LMDV messages. For each landmark entry, joined multicast groups are listed. RD, designed to replace routed, the default routing daemon running RIP protocol, updates kernel routing tables according to the information collected in the course of DV exchange. 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 don’t rely on Linux kernel’s multicast forwarding capability but provide our own packet forwarding engine (PFE) 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 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 receipt of the unicasted packet at the landmarks, PFE broadcast the packet for scoped flooding. PFE on non-landmark nodes re-broadcast every incoming broadcasted packet after checking TTL for scoping and ID for duplicate packet suppression. When rebroadcasting jitter, a random wait, is introduced to reduce conflict between the rebroadcasts. Forwarding the multicast packets to the destination multicast application from PFE is done by loopbacking the packets. Linux supports multicast traffic lookback. M-LANMAR is compatible with any data existing multicast applications using standard socket interface. Intercepting and loopbacking makes M-LANMAR transparent to them.

4. Experiments and Results

4.1. Testbed Configuration

The 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. Our testbed is configured as one multicast source, node 1, and three members that are two hops away. Nodes in dashed circle have same subnet and different circle has different subnet. 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 channel locally reserved to minimize the interference. We developed a multicast traffic generator and collector to see how many packets are eventually delivered to the group members. We also measured M-LANMAR and ODMRP control packet by running tcpdump.

![Figure 2. Testbed Topology](image-url)
4.2. Experiments and Result Analysis

As MLANMAR aims to achieve scalability on large-scale networks, it is difficult for us to show full performance advantage with the experiments performed on a small-sized testbed. We rather focus on the verification of our implementation by showing some benefit still get in the small-sized network.

We compare MLANMAR to two base lines: unicasting and the ODMRP implementation [??]. Since MLANMAR uses unicasting in its two-level data dissemination scheme, it should show comparable results to pure unicasting. (Talk a little about ODMRP implementation) We defined the delivery ratio (DR) to be (the number of packets sent)/(the number of packets arrived at a receiver) and the normalized control overhead (OH) is defined to be (the number of control packets generated)/(the number of packets arrived at a receiver) and measured in the experiments under various traffic conditions. Multicast traffics were generated on node 1 towards node 3, 4 and 5 comprising a receiver team in MLANMAR. Unicast traffics were generated from node 1 to node 3. The results are summarized in Figure 3 & 4. For MLANMAR and ODMRP, the delivery ratio is averaged over 3 receivers. Over various traffic conditions, MLANMAR showed consistent delivery ratio of near 100% similar to unicasting and much higher than that of ODMRP. This result shows that MLANMAR’s using RTS/CTS enabled link-layer unicasting helps reliable packet delivery. In the other hand, as ODMRP uses link-level broadcast, it suffers from data packet losses.

MLANMAR showed higher control OH than ODMRP. In small-scale networks, this is the case but on-demand protocols suffer from high control OH when the number of senders and receivers gets bigger in a large-scale network. MLANMAR’s scalability in terms of control OH is shown in the next section.

5. Simulation of Large number of networks

In the previous section, we verified the implementation of M-LANMAR. Thanks to unicast tunneling mechanism, M-LANMAR shows the higher throughput than ODMRP even in the small scale network. However, if we deploy more nodes, the performance of ODMRP will be improved because of the path redundancy as shown in []. As M-LANMAR targets at the large-scale network, in this section, we show the advantages of M-LANMAR in the large-scale network through simulation study.

We implement 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 [1] 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 5, 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 the 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 team multicast solution. The analysis of flooding shows that the delivery ratio in flooding drops with heavy offered load as shown in Fig. ref{mobility_4pkt}. 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 6 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 control OH of M-LANMAR decreases as the actual offered load increases.

6. Conclusion

In the course of discussion, we intended to share our experience on implementing and validating a scalable multicast protocol for 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. As it targets large-scale networks, validation of the protocol was done using simulation. Our experimental and simulation results showed that by defining a new data delivery paradigm, M-LANMAR achieved improved reliability in data delivery compared to ODMRP.

Reference