Experimental Evaluation of LANMAR, a Scalable Ad-Hoc Routing Protocol

Yeng-Zhong Lee*, Jason Chen*, Xiaoyun Hong†,

Kaixin Xu*, Teresa Breyer*, and Mario Gerla*

Computer Science Department
University of California, Los Angeles, CA 90095
*{yenglee, jvc, xkx, tbreyer, gerla}@.cs.ucla.edu

Department of Computer Science
The University of Alabama, Tuscaloosa, AL, 35487
†{hxy}@cs.ua.edu

Abstract—**Routing protocols for mobile ad-hoc network have been evaluated extensively through simulation because various network conditions can be easily configured, tested, and replicated across different schemes in simulation than in a real system. Recently, some of these schemes have been implemented in academic, industry and defense testbeds. This gives researchers an opportunity to validate their simulation results with actual implementations. In this paper we report the lessons learned from the implementation of LANMAR [1], a scalable routing protocol that was developed at UCLA as part of large-scale ad hoc network architecture for autonomous unattended agents under ONR support. LANMAR is designed to provide efficient, scalable routing in large ad-hoc wireless networks that exhibit group mobility. In this paper we describe the implementation of this protocol in Linux environments and report on experimental results based on this implementation. The results and lessons from these experiments have enriched our understanding of the LANMAR protocol and its interaction with the other layers and the environment, paving the way to protocol refinements and more efficient implementations**

I. INTRODUCTION

Efficient routing is one of the critical issues faced in mobile ad-hoc networks and it has been studied extensively among researchers. Many routing protocols for mobile ad-hoc network have been proposed based on different design principles. Performances of these protocols have also been evaluated extensively through simulations because diversified network conditions can be more easily configured and reproduced in simulation than in real system. Also, the performance of these different protocols can be better compared under identical simulated environments. More recently, in recognition of its significance, researchers and developers have begun to conduct implementation and experimental studies on ad-hoc routing protocols in real environments. The experiences and discoveries from these experiments have greatly deepened and enriched the understandings of ad-hoc routing protocols and contributed to the realization and deployment of mobile ad-hoc networks.

This work was supported by the ONR "MINUTEMAN" project under contract N00014-01-C_0016, and in part by Innovative Concepts "STTR" project under contract 2003-08.

The Landmark Ad-hoc Routing Protocol (LANMAR) [1, 2, 9, 10, 11] is designed to dramatically reduce the number of routing entries needed and routing update overhead in large-scale ad-hoc networks that exhibit group mobility. Theoretical analysis and simulations [2] have been conducted and the protocol design has been evaluated in a variety of simulated scenarios. Simulation results illustrated that for a scenario with 400 nodes, LANMAR achieved 90% packet delivery ratio while a traditional Link State protocol only delivered 50%. However, without implementation and experimentation in real environments, we really do not fully understand how well LANMAR will perform and how it compares to other ad-hoc routing protocols in practice. In this paper, we describe our implementation of LANMAR in the Linux operation system and investigate the performance of the protocol in a real ad-hoc network to further validate the simulation results from previous studies and to improve upon the current protocol.

LANMAR consists of two complementary and cooperating routing schemes: (a) a local "myopic" proactive routing scheme that operates within a limited scope centered at each node and exchanges route information about nodes up to only a few hops; and (b) a "long haul" distance vector routing scheme that propagates the elected landmark of each subnet and the path to it into the entire network. The protocol is well suited to network scenarios where a set of nodes has a commonality of interests and are likely to move as a "logical group". The logical groups are identified using different subnets in LANMAR routing. Each logical group dynamically elects a node as a landmark node that is used to keep track of the group. A packet to a destination outside its local scope will be routed towards the

landmark node corresponding to the destination's group. Routing to all nodes within its local scope uses one of the proactive approaches such as FSR [3], RIP [4], OLSR [5], or TBRPF [6]. As a result, each node essentially keeps two routing tables: local routing table and landmark table which maintain direct routes to near-by destinations and routes to all the landmarks from all the subnets, respectively.

The rest of this paper is organized as follows. In section II, we introduce the implementation of LANMAR and issues concerning the implementation. We then report experiment results evaluating the performance of LANMAR and validating simulation results in section III. Finally, we present our ongoing work in section IV and conclude our paper in section V.

II. IMPLEMENTATION

The implementation of LANMAR is realized as a daemon in user space to minimize changes to the kernel. It concurrently creates and runs separate threads for "local" proactive routing and for "long haul" distance landmark routing as illustrated in Figure 1. Actual routing and forwarding are performed at kernel level to reduce the time and process overhead. We implemented three different link state local scope routing protocols: OLSR, TBRPF, and RIP. In the local routing protocol, only routes to nodes within the predefined scope, say up to hop distance N, are maintained. N is configurable and defines the local scope. And for "long haul" routing, DSDV is currently used to maintain paths to all landmark nodes over the entire network. The LANMAR implementations are called LANMAR-OLSR, LANMAR-TBRPF, and LANMAR-RIP respectively. The two threads read and write to the kernel routing table separately when they need to update the routing entries for either local or landmark nodes (from each thread). To avoid race condition on the kernel routing table, the coordination of the different threads uses a simple semaphore mechanism. The kernel routing table also provides a connection between the two routing components. Particularly, the landmark election calculates election weight, i.e., the number of members in its local scope, directly based on knowledge learned from the kernel routing table, thus avoiding node information exchange between the two threads.

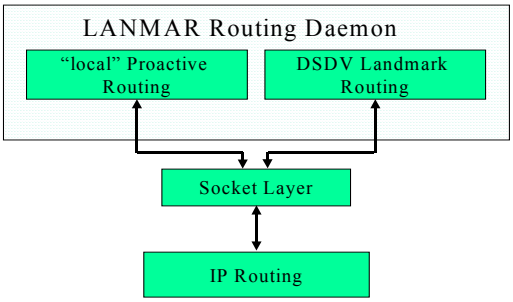

Figure 1: Illustration of LANMAR Implementation structure

The LANMAR routing protocol uses a two-tier address scheme <*SubnetID, HostID*>, where the "*SubnetID*" represents the logical group membership. The landmark hierarchy matches perfectly with the IP subnet hierarchy. Thus, in our implementation, when a landmark entry, which serves as a routing direction towards a subnet, needs to be written into the kernel table, only the subnet address of the landmark node and the next hop node are transferred, along with the corresponding net mask, creating an entry of a subnet in the kernel table.

For outdoors field experiments, we are using ground robots named AmigoBot (that are described and can be ordered at http://www.amigobot.com/) for navigating laptops and iPAQs. We developed an application to send control information over the ad-hoc network to the laptop/iPAQ attached to a robot, and then the laptop/iPAQ forwards the control information to the robot through a serial cable. In order to avoid collision a robot is equipped with six forward and two rear sonars. Figure 2 shows an outdoor experiment with three groups. The logical group of AmigoBots consists of five nodes and there are two groups on a cart: one has two nodes; one has one node. Figure 3 shows a snapshot of routing table of a node on the cart. In this routing table, there are 3 landmarks. Each represents a logical subnet, and the node has 7 neighbors in its local routing table since all nodes in the experiment can hear each other. For example, when one data packet needs to be routed, a node will first consult the kernel routing tables with the corresponding net mask 255.255.255. If a route is found the packet is routed directly otherwise, the destination node is a remote node and the data packet will be routed to the corresponding landmark of the destination's group.

Figure 2: LANMAR test-bed with AmigoBots

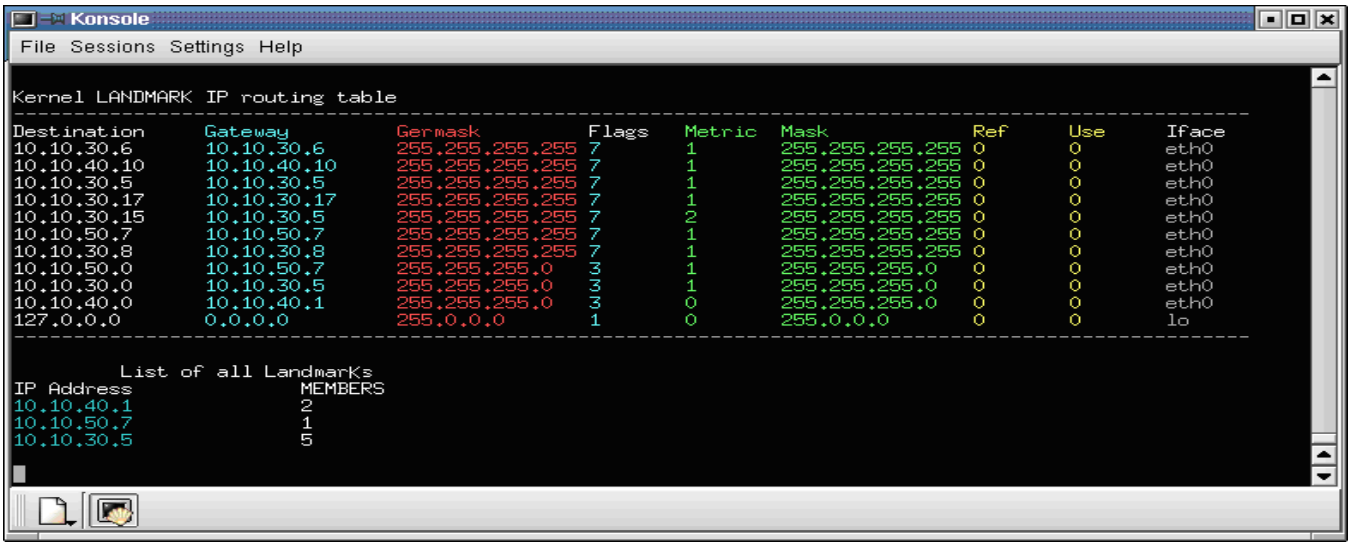

Figure 3: Snapshot of a routing table

III. Performance Evaluation

In this section, we first describe our test-bed environment and simulation results. The main purpose is to validate our implementation of LANMAR and to evaluate performance of LANMAR routing protocol in real ad-hoc networks.

Test-bed environment

Our test-bed consists of ten Dell Pentium III, 650/500MHz Inspiron 4000 laptops and three Compaq 400 MHz Intel PXA250 xscale processor (ARMv5l) H3900 iPAQs equipped with Orinoco 802.11b PCMCIA card with channel rate as 2Mbps and queue size 65536 bytes. The laptops run Mandrake Linux distribution 7 with kernel version 2.4.3 and iPAQs run on ARMLinux [13]. Linux PCMCIA package version 3.2.0 [14] and Orinoco wavelan2-cs driver are used for 802.11b devices and the devices are set to ad-hoc mode. The topology of the test-bed is illustrated in Figure 4. There is one source node X.20.22 and one selected destination node among these nodes, X.20.24, X.30.21, X.30.23, X.40.25, and X.40.23 depending on the number of hops in our experiments. We use "Iperf"[7] to generate UDP based CBR (constant data rate) traffic with a packet size of 500 Bytes on a source to test how many packets are eventually delivered to a destination as well as the throughput, also we use "ping" to gauge end-to-end delay between given nodes. In our experiments, Request-To-Send (RTS) and Clear-To-Send (CTS) control packets are adopted to provide carrier sensing for unicast data packets to overcome the well-known hidden terminal problem. To setup a multi-hop wireless network in a small physical space, a MAC filtering tool "iptables" [12] is used to block packets in the firewall from the designated nodes. The most challenging issue in our experiment is how to emulate mobility. Our approach is to run a script file for MAC filtering on selected nodes. The file emulates mobility by blocking packets from various nodes for a desired period of time. The routing protocols selected for comparison in our study are RIP [4] and LANMAR-RIP, where the routing information refresh interval is set to 1 second for both RIP and LANMAR routing and the scope size of the LANMAR protocol is 2. Each experiment is run for a total of 180 seconds.

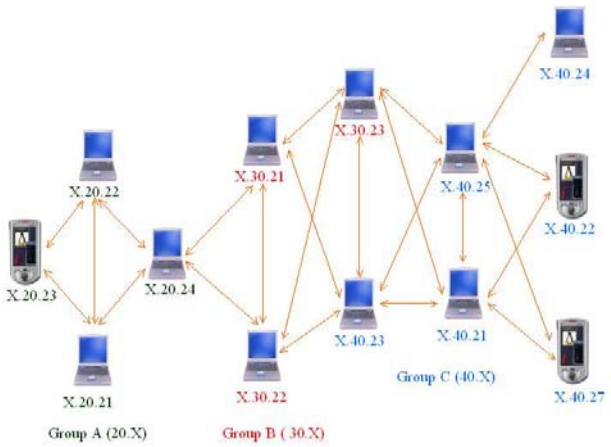

Figure 4: Topology of experiment test-bed

Simulation Configurations

We use Qualnet [8] simulator, a packet level simulator to evaluate the LANMAR routing scheme. Qualnet is a scalable simulation environment for wireless network systems using the parallel discrete-event simulation language PARSEC. The distributed coordination function DCF of IEEE 802.11 is adopted with a channel rate of 2Mbps in all simulations. We developed a MAC filter in

Qualnet to emulate the MAC filtering mentioned above. The routing protocol selected is LANMAR routing that has been implement in the Qualnet simulator; all other configurations are the same as described in the test-bed environment.

Experiments and Results Analysis

The experimental study we conducted focuses mainly on the influence of mobility and the proactive nature of the routing protocol, and also on the advantages of landmark aggregation over groups of mobile nodes. However, as LANMAR aims to achieve scalability on large-scale networks, it is difficult to show full performance advances with the experiments performed on a small-size test-bed. Thus, we focus on the verification of our implementation by showing some benefits that we can still observe in the small-size network. The scalability of LANMAR in large-scale network confirmed throughout the simulation study in [1, 2, 3, 9, 10,11] will not be addressed in this paper. The performance metrics we evaluated include network throughput, latency, packet loss rate, and control overhead. For example, a comparison of simple RIP with LANMAR-RIP would show a reduction of routing overhead and hence an increase in throughput by LANMAR. In our experiments, network conditions vary in terms of the number of hops, link changes, and node mobility.

In our first set of experiments and simulations, source X.20.22 generates CBR traffic at a rate of 1Mbps to a destination among these nodes corresponding to the number of hops: X.20.24, X.30.21, X.30.23, X.40.25, and X.40.23. Figure 5 demonstrates the delivery ratio comparison of RIP and LANMAR-RIP in test-bed experiments and LANMAR in Qualnet over various numbers of hops without mobility. We observe that delivery ratios of all schemes are just about the same and only depend on the number of hops. The delivery ratios decrease as the number of hops increases due to the limited capacity of 2Mbps in the entire network. Similar results, this time in terms of network throughput, are reported in Figure 6.

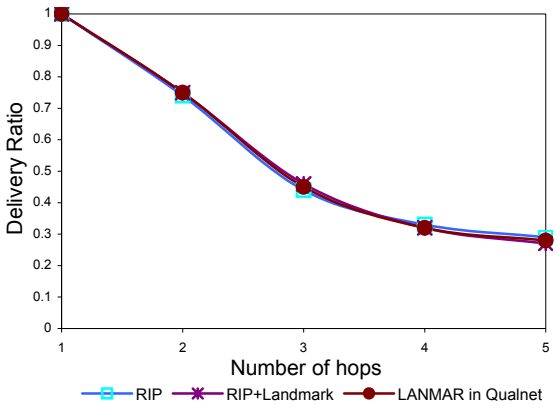

Figure 5: Delivery ratio vs. number of hops

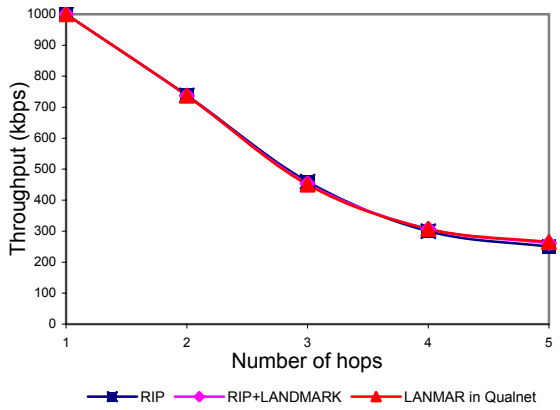

Figure 6: Throughput vs. number of hops

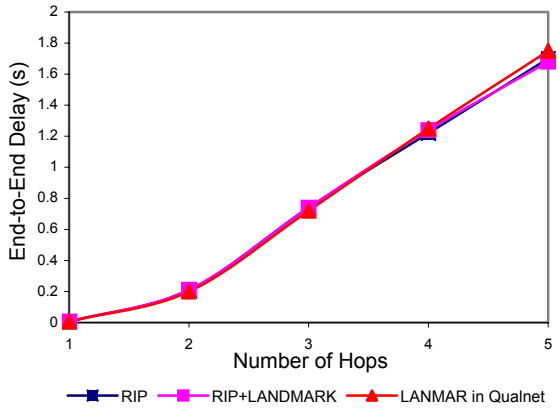

Figure 7: End-to-End delay vs. number of hops

Protocols	Individual nodes	Landmarks	Total
RIP	12	0	**12**
RIP+ LANDMARK	3	3	**6**
LANMAR in Qualnet	3	3	**6**

Table 1: Number of entries in the routing table on node X.20.23

Figure 7 shows average delay as a function of number of the number of hops by using "ping" as described above. As expected the average end-to-end delay increases rapidly with increasing number of hops. This is due to queuing delay on each intermediate node thus leading to greater end-to-end delay the greater the number of hops and shorter delays for fewer hops.

Table 1 gives the number of entries in the routing table on node X.20.23 while experiments and simulations are running. From this table, we can see that using the LANMAR scheme incurs a lower routing table overhead in comparison with other routing schemes even in such a

small network with a small group size. This is because in the LANMAR scheme only one landmark is required to represent all members of its group in a remote node's routing table. For example, in our topology landmark X.40.21 represents group C and only the subnet address of the landmark IP is used instead of all members of group C in node X.20.23 routing table.

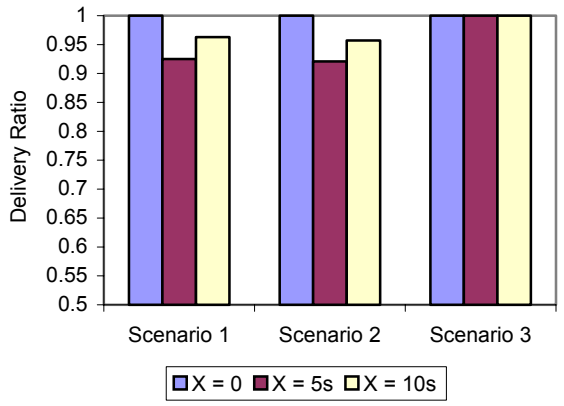

Figure 8: Delivery ratio vs. link changing

In our second set of experiments, the source node X.20.22 generates CBR traffic at a rate of 250Kbps to the destination X.40.24 with intermediate nodes going up and down. The rate of traffic generated was reduced from 1Mbps, in the previous experiment, because we wanted to eliminate loss due to queue overflow. As shown in Figure 6 the minimum throughput of the network is approximately 250Kbps, thus using this rate would ensure a consistent throughput independent of the number of the number of hops. For emulating node up/down, we select two nodes in each scenario that take turns to switch their radio to an unused frequency for a certain time period X (X = 0, 5, 10s). For maintaining connectivity between the source and the destination, the selected nodes will need to properly schedule the dwelling in the unused frequency. In scenario 1, two possible landmarks of an intermediate group C node X.30.21 and X.30.22 are selected. In scenario 2, two possible gateways nodes X.30.23 and X.40.21 are taken. In scenario 3, two possible landmarks of the destination's group node X.40.21 and X.40.22 are selected. Figure 8 shows the delivery ratio of LANMAR-RIP in test-bed experiments. The results demonstrate that the delivery ratio slightly decreases as the number of link changes increases except in scenario 3. This is because the two selected nodes in scenario 1 or 2 are along the path from the source to the destination. In scenario 3, the two selected nodes are not on the path to the destination. Also once the landmark of the destinations group is down, the source and forwarding nodes can still use the landmark to route data packets towards the destination until a new landmark is updated in their routing table. During the experiments we also notice that the average link changing time is 1-1.5s.

In our final set of experiments, we test mobility by applying the script file mentioned above. Figure 9 shows the state of the network and the location of the mobile group. The source node X.20.22 in group A generates CBR traffic at a rate of 250Kbps to the destination X.40.24 in group C. In our script file, the network stays in each step for 10 seconds, and the total experiment time is 30 seconds. In case A, Group A is moving through step A, step B, then step C toward the destination X.40.24. In case B, Group A is moving away from the destination X.40.24 through step C, step B, and then step A. On the left of Figure 10, we show the delivery ratio of each step individually in our experiments. On the right of Figure 10, the average of the delivery ratio in all steps, delivery ratios in case A and case B, are presented. We can see that the delivery ratio in case B is slightly lower than case A and lower than the average. This is because when a group moves towards the destination node, a new shorter path will be established while the old path is still available. On the contrary, when a group moves away from the destination, a new path will not be established until the old path is disconnected. Thus, this additional route update latency incurred while moving away from the destination node leads to the lessened

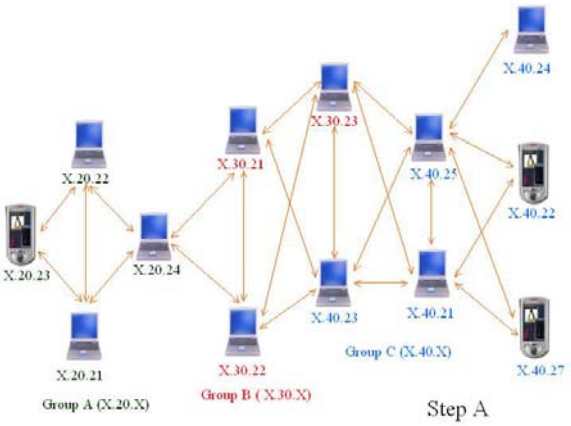

Figure 9 (a): Topology of mobility network on step A

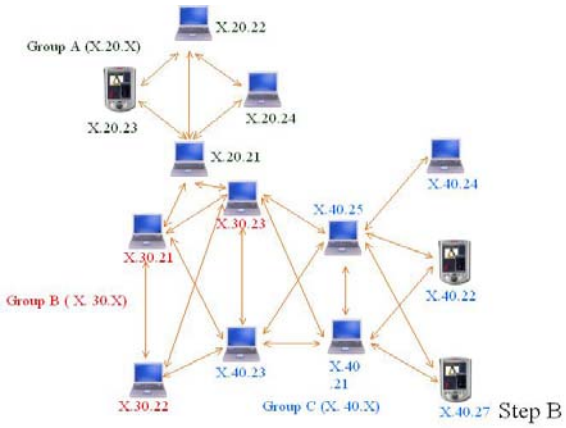

Figure 9 (b): Topology of mobility network on step B

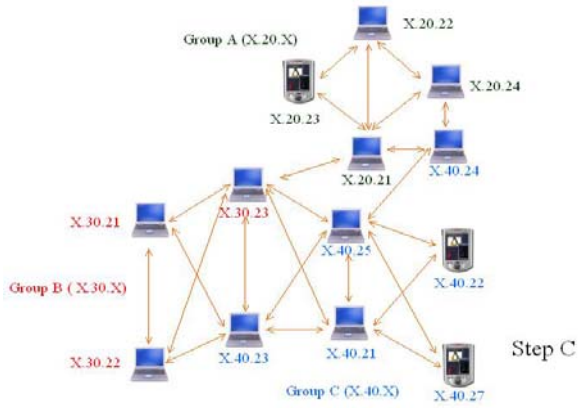

Figure 9 (c): Topology of mobility network on step C

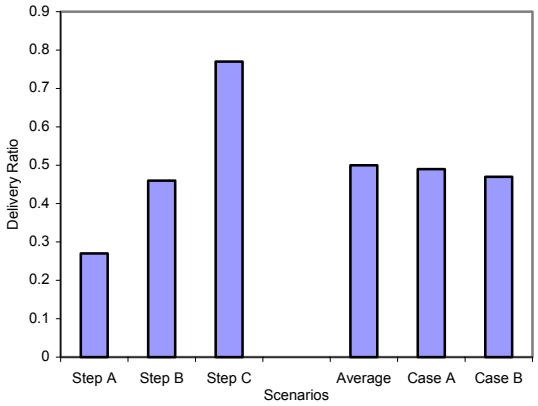

Figure 10: Delivery ratio in mobility network

delivery ratio in case B.

IV. Future Work

With the current state of migration from IPv4 to IPv6 it will be necessary to have an implementation of this protocol that supports IPv6. Also, given the fact that IPv6 accounts for nodes to specify a "group ID" this will allow for better assignment of nodes to specific LANMAR groups to account for mobility. Work has already started towards an IPv6 implementation and experimental results of this implementation are forthcoming. Another area that could be explored in greater depth would be large-scale real world deployment of LANMAR. Large networks of nodes have already been tested in simulations using LANMAR with promising results and given the results we obtained from this study we believe the benefits in large-scale deployment will mirror the simulation results. This will allow for a better evaluation of whether results from a real-world implementation will diverge from the results acquired from simulations or mirror the results attained through simulations. Finally, as stated before, LANMAR can utilize various proactive routing schemes. In this study we only evaluated LANMAR using the RIP protocol. Thus, another area for future work would be to evaluate the efficacy of LANMAR using other routing schemes such as OLSR or TBRPF. Similarly, other methods for discovering and maintaining routes to landmarks will be investigated (e.g., Geo routing; Fisheye routing, etc).

V. Conclusion

Using the current test-bed capabilities, we have been able to conduct real network experiments and to test the accuracy and performance of our LANMAR implementation against simulated results. In particular, we have been able to validate the control overhead efficiency of LANMAR by showing that, even in a small testbed, LANMAR dramatically reduces the amount of overhead associated with more traditional ad-hoc routing protocols. Another interesting property that was noticed for the first time in testbed experiments was the dependence of performance on motion towards the destination or away from destination; namely, the delivery ratio improves when the source moves towards destination as the routing tables are refreshed more frequently. In summary, the lessons learned from the joint testbed and simulation experiments have greatly improved our understanding of the protocol layer interdependencies and will undoubtedly contribute to more efficient designs in the future.

References

[1] G. Pei, M. Gerla and X. Hong, "LANMAR: Landmark Routing for Large Scale Wireless Ad-Hoc Networks with Group Mobility," in *Proceedings of IEEE/ACM MobiHOC 2000*, Boston, MA, Aug. 2000, pp. 11-18.

[2] X. Hong, M. Gerla, Y. Yi, K, Xu, and T. Kwon, "Scalable Ad Hoc Routing in Large, Dense Wireless Networks Using Clustering and Landmarks," in *Proceedings of ICC 2002*, New York City, New York, April 2002.

[3] G. Pei, M. Gerla, and T. -W. Chen, "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks", in *Proceedings of ICC 2000*, New Orleans, LA, Jun. 2000.

[4] C. Hedrick, "Routing Information Protocol," Rutgers University *RFC-1058*, June 1988.

[5] P. Jacquet, P. Muhlethaler, A. Qayyum, A.Laouiti, L.Viennot and T. Clausen, "Optimized Link State Routing Protocol," *draft-ieth-manet-olsr-08.txt*. Internet Draft, IETF MANET Working Group, Feb. 2003.

[6] [6] B. Bellur and R.G. Ogier, "A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks," in *Proceedings IEEE INFOCOM' 99*, New York, March 1999.

[7] Iperp measure tool, available at http://dast.nlanr.net/Projects/Iperf/

[8] Qualnet simulator, available at http://www.qualnet.com.

[9] Xiaoyan Hong, Kaixin Xu, and Mario Gerla, "Scalable Routing Protocols for Mobile Ad Hoc Networks", IEEE Network Magazine, July-Aug, 2002, pp. 11-21.

[10] Kaixin Xu, Xiaoyan Hong, and Mario Gerla, "Landmark Routing in Ad Hoc Networks with Mobile Backbones", in Journal of Parallel and Distributed Computing (JPDC), Special Issues on Ad Hoc Networks, 2002.

[11] Xiaoyan Hong, Kaixin Xu, and Mario Gerla, "Scalable Routing Protocols for Mobile Ad Hoc Networks", IEEE Network Magazine, July-Aug, 2002, pp. 11-21.

[12] iptables, available at http://www.netfilter.org/

[13] ARMLinux, available at http://www.handhelds.org/

[14] Linux PCMCIA driver, available at http://sourceforge.net/projects/pcmcia-cs