MOLSR: Mobile-Agent Based Optimized Link State Routing Protocol

Wassim El-Hajj
Computer Science
American University of Beirut
we07@aub.edu.lb

Ghassen Ben Brahim
Computer Science
Prince Mohammad Bin Fahd University
gbrahim@pmu.edu.sa

Haidar Safa
Computer Science
American University of Beirut
hs33@aub.edu.lb

Maha Akkari
Computer Science
American University of Beirut
ma55@aub.edu.lb

***Abstract*—The popularity of Mobile Ad Hoc networks (MANETs) continues to increase due to the technological advances in wireless radios and wireless networks. Given the unpredictability of the wireless medium, one of the major challenges in MANETs remains to be the efficiency of the underlying routing protocol. The Optimized Link State Routing Protocol (OLSR) is one of the most popular routing protocols being deployed in MANETs. OLSR uses multipoint relay mechanism (MPR) to maintain the topology information at each node. Instead of using blind broadcasting where each node sends a control message to all its neighbors, and consequently every neighbor forwards the message to its neighbors, MPR selects a small set of nodes to do the broadcasting such that all network nodes are covered. In this paper, we propose improving further the performance of OLSR by introducing the concept of Mobile Agent (MA) when maintaining the MANET topology information, resulting in MA-based OLSR (MOLSR). In MOLSR, MA will substitute MPR and will rely on message unicast instead of broadcast, a major improvement to current OLSR in terms of traffic overhead. In MOLSR, every node creates and launches a MA that intelligently travels the network and returns with the full topology. In case of an MA loss, MOLSR automatically recovers the MA and makes sure it covers the whole network. Extensive simulation results show that MOLSR outperforms OLSR in various areas, namely the total control messages, network utilization, and reliable message transfer.**

***Keywords*—MANET, Routing, OLSR, MPR, Mobile Agents**

I. INTRODUCTION

The popularity of MANETs [1] is continuously increasing due to the advancements in wireless radios and mobile devices. Such popularity also comes from the fact that MANETs allow people to easily go online and connect with each other without the need for fixed infrastructure. Such communication can be done using direct one-to-one communication between users using public or private hotspots or using smart radios (such as cognitive radios). One of the major challenges in MANETs is maintaining topology information given the unpredictability of the wireless medium.

Link State routing protocols [2] have been traditionally used in MANETs to maintain global network information at each node. Such global information is gathered by exchanging hello messages between nodes. Each node starts by sending a hello message to its neighbors. Each neighbor then forwards the message to its neighbors and so on. The process continues until the hello message reaches all nodes in the network. The drawback of such flooding mechanism is the high message overhead it entails.

To reduce the message overhead, an enhancement to the classical link state algorithm called Optimized Link State Routing Protocol (OLSR) [2] was proposed. The main optimization added to OLSR lies in the flooding mechanism via selecting certain nodes called multipoint relays (MPRs) [4] to forward broadcast messages. Each node chooses its MPR set from its 1-hop neighbors. The MPR set is selected such that it can reach all symmetric strict 2-hop neighbors i.e. 2-hop neighbors which are not the node itself or a neighbor of the node, and in addition are neighbor of a neighbor. The one-hop and two-hop neighbor sets are gathered through periodic exchange of HELLO messages. This smart flooding technique significantly reduces the message overhead as compared to the classical flooding mechanism.

In this paper, we propose a novel alternative mechanism to MPR: MOLSR. MOLSR uses Mobile Agents (MA) [3] to collect routing information across the whole network. An MA is a self-learning program, capable of traveling across nodes, collecting information and changing its behavior according to the new information it acquires. By the time the MA returns to its launcher (returns home), it will contain the complete topology information. Each node in the network creates and launches a MA that traverses the network, collects topology information and returns to its home carrying the full network topology. Hence, every network node is always aware of the complete network topology. A major advantage of using MAs over MPR is the usage of unicast more than broadcast. Moreover, the MA is equipped with smart algorithms to recover from failures, return back to the node that launched it efficiently, and seize traversing the whole network when it collects all topology information. The conducted simulations provide a proof of the efficiency of MOLSR when compared to OLSR.

The rest of the paper is organized as follows. Section 2 overviews the work related to optimizing OLSR. Section 3

describes the details of MOLSR. Section 4 details the experimental set up and simulation results. Section 5 concludes the paper and presents future work.

II. RELATED WORK

In what follows, we overview the work related to optimizing OLSR. Xue et al. [5] reduced the overhead in OLSR by utilizing three optimizations that rebuilt the format and operating tactic of HELLO and control messages. First, they reduced the size of HELLO messages by comparing the link-states of neighbor set with the existing ones so that only changed links and multipoint relays are transmitted. Second, they decreased the number of TC messages by having MPRs calculate the first TC message rather than every node. Third, they eliminated redundant TC messages by having only one node send link-state information to its MPR rather than all nodes that have the same MPR selector. However, their approach studied a special case of OLSR which does not take redundancy into consideration despite the fact that OLSR protocol includes many schemes which have a variety of information redundancies. These include the number of MPRs and other link-state information encapsulated into TC messages. In this case, the overhead problem becomes much more important.

Enneya et al. [6] proposed enhancements to the original OLSR protocol that adapts to high mobility. The approach relied on a new quantitative measurement of node mobility at each node in the MANET. The measurement was related to the link status change in the vicinity of the communication range. They used the new measurement in three new proposed versions of OLSR. Each version depends on a new criterion for MPR selection. The first criterion was just the mobility of one-hop nodes. The other two criteria are based on the mobility of both one-hop and two-hop nodes. Their results showed that their method required more routing packets to establish and maintain routes in the network.

Toutouh et al. [7] used a series of meta-heuristic algorithms (particle swarm optimization, differential evolution, genetic algorithm, and simulated annealing) to search for optimal configurations of OLSR. The authors evaluated the configurations in a set of realistic VANET scenarios. Their results showed that their tuned OLSR configurations resulted in better quality of service (QoS) than the standard version [2].

Belhassen et al. [8] introduced a novel version of OLSR called the Cartography Enhanced Optimized Link State Routing Protocol (CE-OLSR). Cartography is the study and practice of making maps. Their approach utilizes a cartography gathering scheme to enhance the tracking of node movements in MANETs. The cartography is richer than the traditional topology information. CE-OLSR uses the cartography to ensure better responsiveness in coping with node mobility. Their simulations showed that, at certain speeds, CE-OLSR achieved better route validity, throughput and a smaller average end-to-end delay than OLSR. However the throughput of CE-OLSR is always affected by mobility.

Adoni and Joshi [9] introduced multipath OLSR in order to optimize the energy of the nodes in the network. When choosing paths between two distant nodes at a certain time, those containing higher-energy intermediate nodes were favored. Their simulation results showed a 10 to 25% improvement in the number of nodes alive. There was also a reduction in average end-to-end delay. However, as anticipated, their method increased normalized routing overheads.

Yelemou et al. [10] presented a method to improve OLSR quality of service. They relied on the binary error rate metric in the MPR selection algorithm and route computation. Their results showed that their approach provided a better packet delivery ratio (PDR) than original OLSR. A major drawback in this approach is that only stationary nodes are considered. In case of mobility, standard OLSR and BER-based one have similar performance.

Yang et al. in [11] proposed improving the performance of OLSR by tuning and adjusting OLSR configuration parameters, namely: refresh interval timers and Hello message intervals. Following a different approach, Shahram et al. in [12] proposed improving OLSR performance by eliminating the unnecessary loops. In [13], Omar et al. proposed using clustering techniques to cope with the node mobility and Ant Colony optimization technique for MPR selection. This technique showed its effectiveness in improving the end-to-end message transfer delay as well as the reduction in the message control overhead.

Our approach differs from previous approaches by using Mobile Agent to collect topology information across the whole network. Our approach is built-in OLSR and not and add-on, in the sense that the other approaches either slightly update the inner working of OLSR or add extra modules, while MOLSR integrates completely with OLSR without having a need to any add-ons. Additionally, all previous algorithms use the broadcast mechanism in the topology information collection phase while MOLSR uses unicast. In the next section, we discuss the details of MOLSR.

III. MOLSR DESIGN

MOLSR was designed with the option of turning the MPR algorithm ON and OFF. When MPR option is ON MOLSR is equivalent to standard OLSR, however when MPR option is turned OFF the MA capability kicks in. The goal of such design insures that minimal disturbance is introduced to a pretty stable standard, OLSR. Figure 1 shows a high level architecture of MOLSR. In this figure, we could easily see that either the left side branch could run (i.e. Phase 1-OLSR followed by phase 2-MPR followed by phase 3 –OLSR) which is the standard OLSR, or the right side of the branch could run (i.e. Phase 1-OLSR followed by phase 2-MA followed by phase 3 –OLSR) which is the Mobile Agent OLSR.

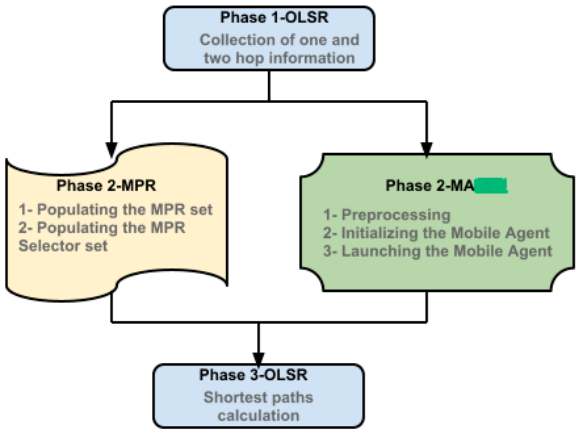

Figure 1. MOLSR high level architecture

MOLSR uses mobile agents to maintain the topology information. MA carries information with it as it travels across the network. This information is summarized in Table 1. Upon the gathering of the two-hop information using OLSR's normal HELLO messages, the MA executes the following three steps: (1) Preprocessing, (2) Initialization, and (3) Launching. These are captured in Phase 2-MA of Fig 1 (right branch). In the preprocessing phase every node decides whether it needs to launch a mobile agent to collect topology information, or not. A node launches a MA only if it does not have the full topology. In the initialization phase, every node that does not have the full topology prepares its mobile agent by setting the appropriate parameters. In the launching phase the MA is launched in the network and is expected to return to its launcher with the full topology information. Each of the three steps is covered in detail in the following sections.

Table I. Information carried by MA as it travels over the network

- ***IP:*** IP of the Launcher
- ***θ:*** The set of one-hop neighbors of the Launcher
- ***μ***: The Set of paths leading to the target nodes (nodes that MA is most likely to visit and gather information from). ***μ*** is represented as a set of pairs (***x***, ***y***) where ***y*** is a target node and ***x*** is the intermediate node on the path to ***y***.
- ***Y:*** Next node that MA will visit and gather topology information from
- **X:** Intermediate node on the path to ***Y***
- ***P:*** Path that MA has taken so far
- ***M:*** Mode of MA. It can be either ACTIVE or RETURNING. ACTIVE is when a MA is collecting topology information. RETURNING is when a MA has finished collecting topology information and is returning home to the node who created it

A. PREPROCESSING

Each node α computes the set of strict two-hop neighbors $\varphi = \theta' - \theta$ where θ is the one-hop neighbor set and θ' is the two-hop neighbor set of α. θ' and θ are known from the first phase (phase 1 – OLSR). If φ is empty then all the nodes in the network are in the θ of α. Hence, α already has the full topology and it does not need to create a MA.

If φ is not empty, then there are nodes in the network for which α does not have topology information for. These nodes are all $x \in \varphi$. Hence α needs to initialize and launch a MA to start gathering network topology.

To clarify this phase, consider the example network in Figure 2. The strict two-hop neighbors of Node 1 is an empty set. Therefore, Node 1 has the full topology and will not create a MA. On the other hand, node 3 has $\theta = \{1\}$ and $\theta' = \{2,4\}$. Thus, $\varphi = \theta' - \theta = \{2,4\}$. The distinct two-hop set of Node 3 is non-empty. This means it does not have the full topology. More specifically, nodes 2 and 4 need to be visited. Therefore, node 3 will initialize its MA (Step 2) and then launch it (Step 3).

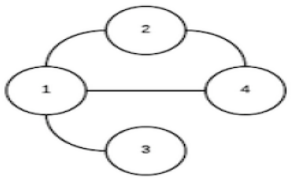

Figure 2. 4-Node MANET network

B. INITIALIZATION

Given $\varphi \neq \varnothing$, a node α will initialize the information carried by its mobile agent (variables in Table 1) to the following values:

Table II. MA Initialization

IP = Ip of ***α***
θ = Set of one-hop neighbors of ***α***
μ= Φ_{α}
Y = y_0
X = x_0
P =*{ **α** }*
M = ACTIVE

The MA is now ready for the launching step, which is covered next.

C. LAUNCHING

The behavior of a MA is determined by one of its two modes: ACTIVE or RETURNING. An ACTIVE agent executes the algorithm depicted in the flowchart of Figure 3 while a RETURNING agent executes the *GoHome* algorithm as depicted in the flowchart of Figure 4. We will discuss the behavior of a MA in both modes and provide an example to illustrate the execution of the MA algorithm in the following section.

Case 1: ACTIVE mode Agent

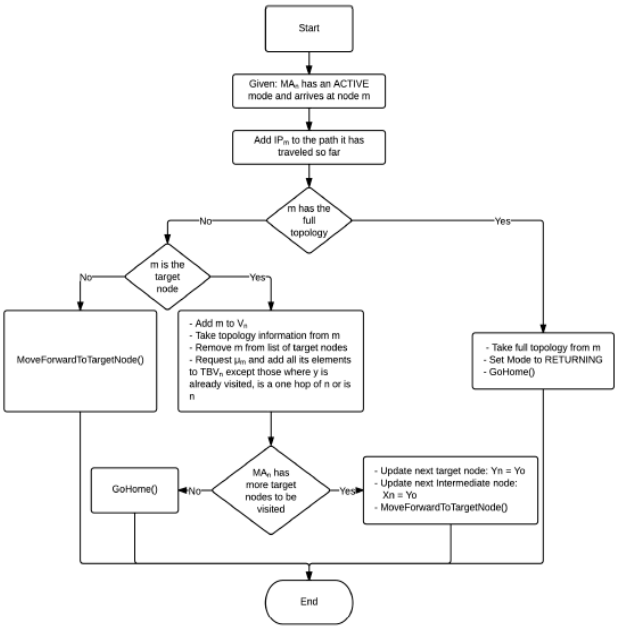

Figure 3. Behavior of MA_n in ACTIVE mode at node m

When MA_n arrives at node m, it adds m to the list of nodes it has visited so far then checks if m has the full topology. If Node m has the full topology, MA_n takes the full topology from m, sets its mode to RETURNING then goes back to originator node by executing the *GoHome* algorithm as depicted in the flowchart of Figure 4. Otherwise, MA_n continues gathering information about full topology and checks whether node m is the target node. If node m is the target node, MA_n heads back to originator. Otherwise, m is an intermediate

node and hence MA_n executes *MoveForwardToTargetNode* algorithm as depicted in flowchart of Figure 5.

Case 2: RETURNING mode Agent

This is the case where MA_n has set its mode to RETURNING because it has finished collecting the topology information and wants to return to its originator. When MA_n arrives at node m, if m is not the originator, MA_n checks whether m is in the two-hop set of n. In the event that m is in the two-hop set of n, MA_n goes to node L, where L is the intermediate node between n and m. Otherwise, MA_n calls the *FindPathToOriginator* procedure in order to find a path to its originator. The *FindPathToOriginator* procedure attempts to find a path to the originating node from each previously visited node by revisiting the previously visited nodes in the same order they were visited. In the next section we illustrate the behavior of MA in a simple MANET network topology.

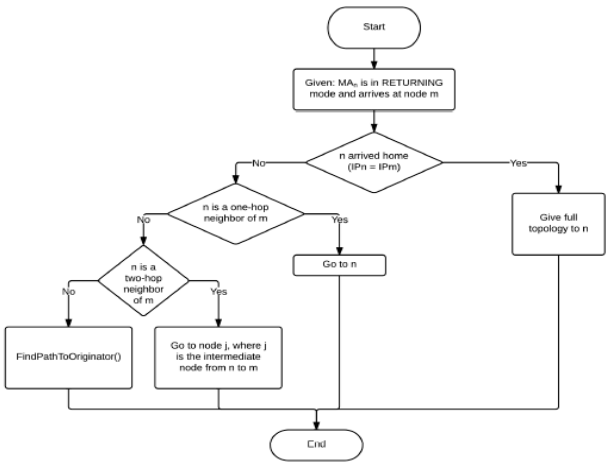

Figure 4. Behavior of MAn in RETURNING mode at node m

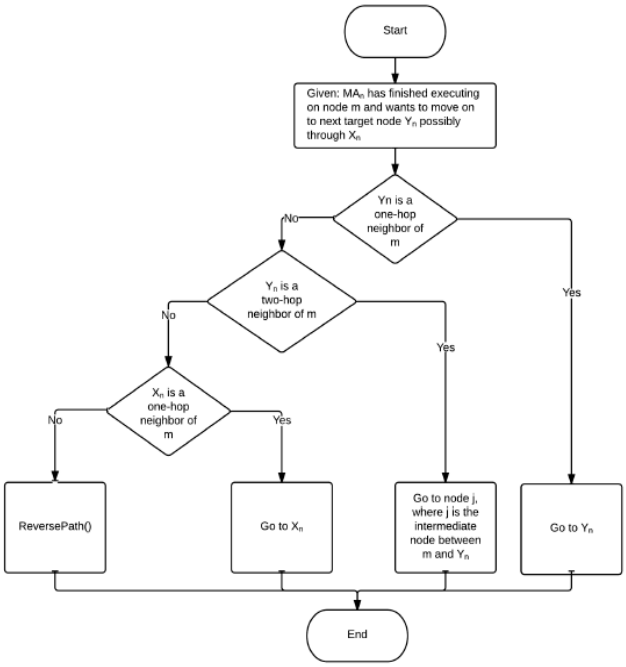

Figure 5. MoveForwardToTargetNode Flowchart

D. Case Scenario

Consider the network in Figure 6 and let us walk through the behavior of node 1. The bold black arrows indicate the path taken by the MA while traversing the network. The dotted red arrows represent the way back.

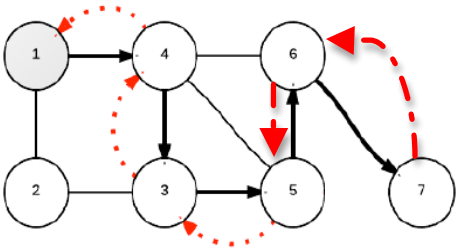

Figure 6. Execution of MA1

Node 1 begins in the preprocessing step, by computing the set of strict two-hop neighbors φ. As a reminder, φ = θ' - θ where θ is the set of one-hops and θ' is the set of two-hops. In this example, θ = {2,4} and θ' = {3,5,6}. Therefore, φ = {3,5,6} - {2,4} = {3,5, 6}. During the preprocessing step, node A creates an MA to explore nodes 3, 5, and 6.

In step two, Node 1 initializes the information carried by the MA (Table 1). The initialized values for A are as follows: IP = 1; θ = {2,4}; μ = { (4,3), (4, 5), (4,6) }; Y = 3; X = 4; P = {1}; M = ACTIVE

Now the MA is in the execution phase (see Fig.3), and follows the flowchart in Figure 5. The MA will attempt to reach its next target, node 3. It is not currently at its target so it executes algorithm *MoveForwardTotarget* (Fig. 5) which tells the MA to go to node 4. Node 4 does not have the full topology and it is not the target. Again the MA executes *MoveForwardTotarget* which takes it to its target, node 3.

Node 3 does not have the full topology but it is the target node. According to the execution algorithm (see Fig. 5), the MA requests the set of strict two-hop neighbors φ of node 3. Node 3 knows its one-hop and two-hop neighbors from the exchanging of OLSR's HELLO messages. Therefore, node 3 provides the value of φ = {1,4,5,6} - {2,4,5} = {1,6}. Node 3 is added to the set of previously visited nodes and the pair (4,3) is removed from the set of potential paths. The elements of φ and their intermediary nodes are added to the set of potential paths, with the exception of node 1 because it is the originator. Thus μ = { (5,6), (4, 5), (4,6) }.

The next target node is updated to be node 6. The *MoveForwardTotarget* procedure determines that node 5 is an intermediate node. Therefore, the MA travels to node 5. From this point, the execution phase repeats. Node 5 does not have the full topology so the MA continues onto node 6. This is the target node so the set of strict two-hop neighbors is requested. Node 6 returns φ = {1,3,4,5} - {4,5,7} = {1,3}. The pair (5,6) is removed from μ. Neither of the distinct two-hops of node 6 are appended because 1 is the originator node and 3 is a previously visited node. Thus μ = { (6,7), (4, 5), (4,6) }.

The next target node is updated to be node 7. The *MoveForwardTotarget* procedure is executed and finds that 7 is one hop away from node 6. The MA travels directly to node 7. This is the target node and node 7 does not have the full topology. The MA requests the distinct two-hop neighbor set. Node 7 returns φ = {4,5} - {6} = {4, 5}. The pair (6,7) is removed from μ. Neither of the distinct two-hops of node 7 are

appended because nodes 4 and 5 are previously visited. Also nodes 5 and 6 were already visited, which makes μ empty.

There is no next target node because μ is empty. This causes the MA to switch from ACTIVE to RETURNING mode triggering the execution of algorithm in Figure 4 to deliver the MA to the originating node 1 and hence provide the full topology to node 1.

IV. Performance Results

A. Simulation setup

In this section, we give some experimental results to illustrate the performance of the proposed protocol which we compare to that of the standard OLSR. Our set of experiments was conducted using NS-3 along with LUA scripting language to simulate MA nodes. In our setup, nodes were randomly distributed on a 3000m by 3000m surface area. The network size considered ranges from 30 to 200 nodes. The network type considered was 802.11b with 250m transmission range. Constant speed propagation delay was considered.

Initially, we run standard OLSR to collect the 1 and 2-hop neighbor information, then we disable MPR and enable MOLSR. MOLSR kicks in with the creation of MAs. MAs will do some preprocessing and possibly go into the initialization phase. After the initialization phase, every MA waits for a random time between 0 and 50 ms to avoid transmission collisions then launches itself in the network. To ensure reliable delivery of the Mobile Agents, we have implemented our own acknowledgment mechanism. We analyzed the performance of MOLSR with respect to the following 5 metrics: (1) Number of messages sent, (2) Message delivery rate (MDR), (3) Traffic within the network, (4) Delay, and (5) Utilization.

B. Results

The first experiment consists of studying the performance of MOLSR in terms of the number of messages sent. In Figure 6 we observe that MOLSR outperforms OLSR in the **total number of messages** exchanged on the network. The number of messages sent decreased by approximately 45%. This performance enhancement obtained with MOLSR is due to the fact that MOLSR uses the unicast mechanism while OLSR uses broadcast.

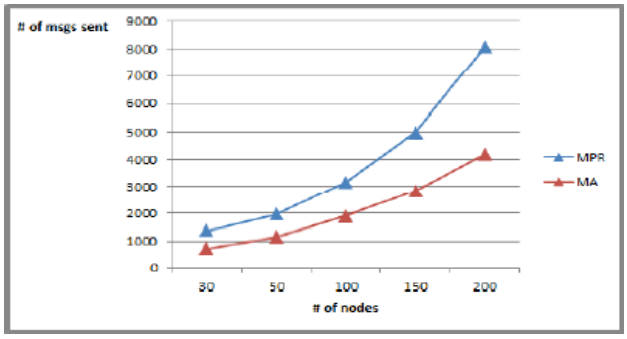

Figure 6. Number of msg sent vs. network size

In the next experiment, we compare the reliability of MOLSR in terms of successful message transmission. Figure 7 shows an increase of 3.6% of MOLSR over OLSR in terms of the **rate of successfully received messages** with respect to the total number of message that were sent. MDR is affected by the traffic load on the network. Since MOLSR is unicast and MPR is broadcast, MOLSR injects much less traffic load on the network and hence decreases the chances of collisions and therefore increases the probability of a successful transmission.

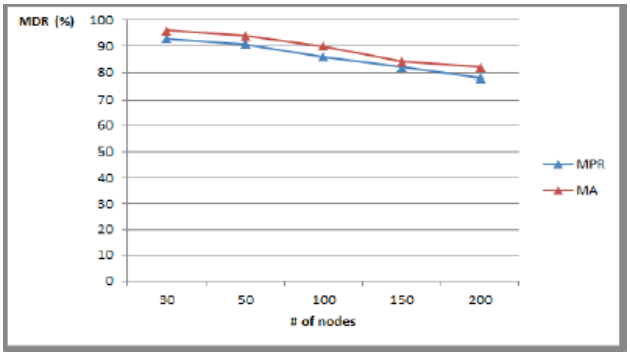

Figure 7. MDR vs. network size

The next experiment consists of studying the performance of MOLSR in terms of the total traffic size. Figure 8 shows that the **total traffic size** in bytes when using MOLSR is within an acceptable range, though slightly higher compared to that of MPR. There was an 8% increase in the traffic size. This is due to the increase in the individual packet sizes. MPR message contains less topology information, while the MA is a file that carries the topology information that is being gathered on the network along with the code that has to be executed on the network nodes. It is worth mentioning here that the performance was achieved after implementing a compression/ decompression algorithm to compress/decompress the MA files. The algorithm begins by initializing a dictionary to all possible input characters. It then scans the input for the longest string that is in the dictionary. Finally, it sends the index of that string to the output and adds that string plus the next character to the dictionary.

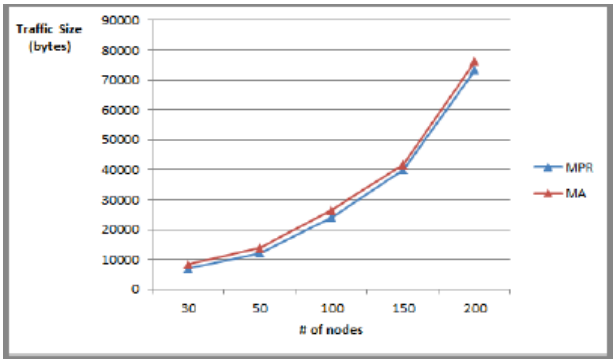

Figure 8. Traffic size (in Bytes) vs. network size

The next experiment consists of studying the performance of MOLSR in terms of the average latency. Figure 9 shows that across the entire simulation the **average latency** was increased by 7%. This is due to the extra traffic induced in the network to insure reliable packets delivery, which was a MOLSR design decision. From the other side, MPR did not plan for mechanisms to handle packet losses because it periodically re-broadcast packets. There is always a tradeoff between improving the reliability of message transfer and latency.

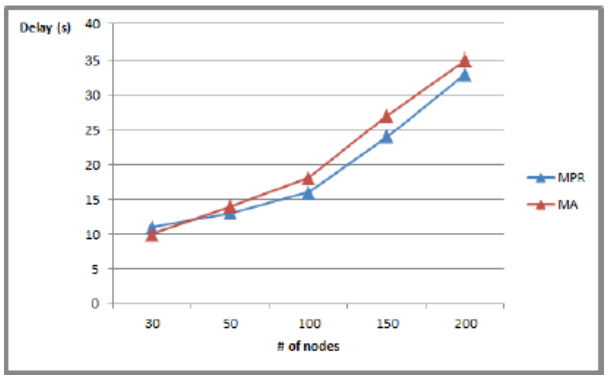

Figure 9. Message delay vs. network size

The last experiment focuses on studying the performance of MOLSR in terms of the utilization. In Figure 10, we observe that MOLSR decreases the node **utilization** by 34%. This is also due to the fact that MOLSR uses the unicast mechanism while in OLSR a node uses broadcast. In MOLSR a node only receives packets that concerns it and are destined for it. While in OLSR a node receives packets that have nothing to do with it and will need to rebroadcast it.

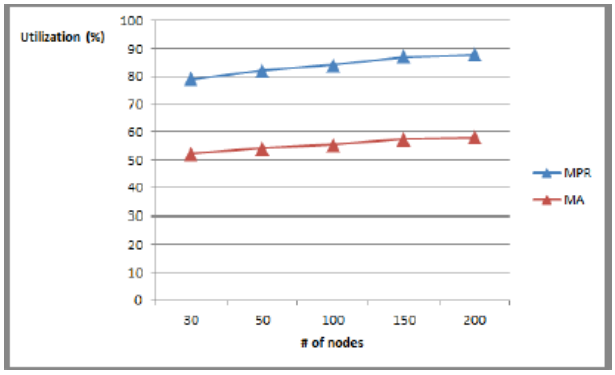

Figure 10. Utilization vs. network size

V. Conclusion and Future Work

In our work, we have introduced MOLSR—a new Mobile Agent paradigm as an improvement over OLSR. While OLSR broadcasts multipoint relays (MPR) to maintain the topology information, our approach uses Mobile Agents to maintain topology information. We have shown that using our approach, we obtain a significant reduction in the number of messages exchanged over the network. This is due to the fact that our approach avoids flooding by using unicast as opposed to broadcast that is used by MPR. Our simulation results showed a considerable decrease in utilization, time and an increase in throughput. However our approach did slightly increase the delay and the traffic size. In our future work, we propose working on reducing further the size of the LUA code by applying a better compression/decompression algorithm. Then we plan to make advantage of the gathered partial network topology for better route selection during the network traversal. This will reduce the latency, network congestion, and number of exchanged messages.

ACKNOWLEDGMENT

This work was supported in part by a grant from the University Research Board of the American University of Beirut, Lebanon.

Reference

[1] Corson,S.,and Macker,J.,"Mobile ad hoc networking (manet): routing protocol performance issues and evaluations considerations", in IETF RFC (January 1999), no. 2501.

[2] RFC 3626. "Optimized link state routing protocol (olsr)", October 2003. Viennot and A. Laouiti,"Multipoint Relaying for Flooding Broadcast Message in Mobile Wireless Networks", Proc. 35th Ann. Hawaii Int'l Conf. System Sciences (HICSS), vol. 9, p. 298, Jan. 2002.

[3] Y. Aridor and M. Oshima, "Infrastructure for Mobile Agents: Requirements and Design", Proc. Second Int'l Workshop Mobile Agents, LNCS 1477, Springer-Verlag, New York, Vol. 1477, 1998, pp. 38-49.

[4] Viennot and A. Laouiti, "Multipoint Relaying for Flooding Broadcast Message in Mobile Wireless Networks", Proc. 35th Ann. Hawaii Int'l Conf. System Sciences (HICSS), vol. 9, p. 298, Jan. 2002.

[5] Y. Xue, H. Jiang, H. Hu, "Optimization on OLSR protocol for lower routing verhead", RSKT'08 Proceedings of the 3rd international conference on Rough sets and knowledge technology, Berlin, Germany, 2008, pp. 723-730.

[6] N. Enneya, K.Oudidi and M. Elkoutbi, "Enhancing Delay in MANET Using OLSR Protocol," Int'l J. of Communications, Network and System Sciences, Vol. 2 No. 5, 2009, pp. 392-399.

[7] J. Toutouh, J. Garcia-Nieto and E. Alba, "Intelligent OLSR Routing Protocol Optimization for VANETs", IEEE Transactions on Vehicular Technology, May 2012, Vol. 61, Issue 4, pp. 1884-1894.

[8] Belhassen, A. Belghith and M.A. Abid, "Performance evaluation of a cartography enhanced OLSR for mobile multi-hop ad hoc networks", Wireless Advanced (WiAd), June 2011, pp. 149-155

[9] Kirti Aniruddha Adoni and Radhika D. Joshi, "Optimization of Energy Consumption for OLSR Routing Protocol in MANET", International Journal of Wireless & Mobile Networks (IJWMN) Vol. 4, No. 1, February 2012, pp. 251-262.

[10] T. Yelemou, P. Meseure and A.M. Poussard, "A new BER-based approach to improve OLSR protocol", Eighth International Conference on Wireless and Optical Communications Networks (WOCN), May 2011, pp. 1-5.

[11] Yang Huang, Saleem Bhatti, Daryl Parker, "Tuning OLSR", The 17th Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC'06).

[12] Shahram Behzad Reza Fotohi, and Shahram Jamali, "Improvement over the OLSR Routing Protocol in Mobile Ad Hoc Networks by Eliminating the Unnecessary Loops", International Journal of Information Technology and Computer Science, 2013, 06, 16-22.

[13] Omar Abdel Wahaba, Hadi Otrokb, Azzam Mourada, "VANET QoS-OLSR: QoS-based clustering protocol for Vehicular Ad hoc Networks", Computer Communications, vol. 36, pp. 1422-1435, 7/15/ 2013.