The Optimized Link State Routing Protocol Evaluation through Experiments and Simulation

Thomas Heide Clausen, Gitte Hansen, Lars Christensen
Gerd Behrmann

Mindpass Center for Distributed Systems
Aalborg University, Fredrik Bajers Vej 7E, DK-9220 Aalborg, Denmark
T.clausen@computer.org, {gitte,larsch,behrmann}@cs.auc.dk

Abstract

In this paper, we describe the Optimized Link State Routing Protocol (OLSR) [1] for Mobile Ad-hoc NETworks (MANETs) and the evaluation of this protocol through experiments and simulations. In particular, we emphasize the practical tests and intensive simulations, which have been used in guiding and evaluating the design of the protocol, and which have been a key to identifying both problems and solutions.

OLSR is a proactive link-state routing protocol, employing periodic message exchange for updating topological information in each node in the network. I.e. topological information is flooded to all nodes in the network.

Conceptually, OLSR contains three elements: Mechanisms for neighbor sensing based on periodic exchange of HELLO messages within a node's neighborhood. Generic mechanisms for efficient flooding of control traffic into the network employing the concept of *multipoint relays* (MPRs) [5] for a significant reduction of duplicate retransmissions during the flooding process. And a specification of a set of control-messages providing each node with sufficient topological information to be able to compute an optimal route to each destination in the network using any shortest-path algorithm.

Experimental work, running a test-network of laptops with IEEE 802.11 wireless cards, revealed interesting properties. While the protocol, as originally specified, works quite well, it was found, that enforcing "jitter" on the interval between the periodic exchange of control messages in OLSR and piggybacking said control messages into a single packet, significantly reduced the number of messages lost due to collisions. It was also observed, that under certain conditions a "naive" neighbor sensing mechanism was insufficient: a bad link between two nodes (e.g. when two nodes are on the edge of radio range) might on occasion transmit a HELLO message in both directions (hence enabling the link for routing), while not being able to sustain continuous traffic. This would result in "route-flapping" and temporary loss of connectivity.

With the experimental results as basis, we have been deploying simulations to reveal the impact of the various algorithmic improvements, described above.

1. Introduction

A mobile ad-hoc network (MANET) is a collection of nodes, which are able to connect on a wireless medium forming an arbitrary and dynamic network with "wireless links". That is, over time the links between the nodes may change due to node mobility, nodes may disappear and new nodes appear in the network. The physical size of a MANET is expected to be larger than the radio range of the wireless interfaces, thus for any two nodes in the network to be able to communicate, routing of traffic through a multi-hop path is necessary.

In addition to the challenge presented by routing in traditional wired networks, a routing protocol for MANETs must be able to respond to a high degree of topological changes in the network and still maintain stable routing. Also, the peculiarities of wireless interfaces (such as the inherent differences from wired media in medium access characteristics as well as the broadcast nature), must be taken into account. Finally, the fact that the available bandwidth on a wireless link currently is orders of magnitude smaller than that available in wired networks, requires that a protocol be carefully designed to reduce the amount of control traffic generated.

Several protocols exist, addressing the problems of routing in mobile ad-hoc networks. Such protocols are, traditionally, divided into two classes, depending on when a node acquires a route to a destination. *Reactive* protocols are characterized by nodes acquiring and maintaining routes on-demand. I.e. a route to a destination is not acquired by a node before a packet to that destination appears in the node, delaying the transmission of a packet until a route is available (or until it is found that a route is unobtainable). Examples of reactive protocols include the "Ad Hoc On Demand Distance Vector Routing Protocol" (AODV) [2] and "Dynamic Source Routing" (DSR) [3]. *Proactive* protocols are characterized by all nodes maintaining routes to all destinations in the network at all times. Thus, using a proactive protocol, a node is immediately able to route (or drop) a packet. Examples of proactive protocols include the "Mobile Mesh Routing Protocol" (MMESH) [4] as well as the "Optimized Link State Routing Protocol" (OLSR) [1], as described in this paper.

1.1. Paper outline

The remainder of this paper will be organized as follows: in section 2, we describe the OLSR protocol as a flexible proactive routing protocol for MANETs, emphasizing the three independent components making up the protocol. In section 3 we describe our experiences from conducting practical experiments with the protocol. This includes some inconveniences encountered, and the measures taken to counter these. Section 4 describes our simulation results, further uncovering attributes of the protocol. Section 5 will discuss our results, and the paper is concluded in section 6.

2. The Optimized Link State Routing Protocol

OLSR [1] is a proactive, link-state routing protocol, employing periodic message exchange to update topological information in each node in the network. That is, topological information is flooded to all nodes in the network, providing routes immediately available with a constant, low control traffic overhead - regardless of data load and node mobility causing link breakage.

Conceptually, OLSR contains three generic elements: a mechanism for neighbor sensing, a mechanism for efficient flooding of control traffic, and a specification of how to select and diffuse sufficient topological information in the network in order to provide optimal routes. These elements are described in details in the following.

2.1. Neighbor Sensing

Neighbor sensing is the process through which a node detects changes to its neighborhood. This section will describe in detail the neighbor sensing mechanisms included in OLSR.

2.1.1. Definitions

A node *a* is said to be a *neighbor* of another node *b* if there exists a direct link between the two nodes *a* and *b*, allowing data to be transmitted in either or both directions of the link. The link between two neighboring nodes can further be characterized by the direction in which communication is possible. Thus, if communication is possible from node *a* to node *b* (i.e. traffic from node *a* can be received at node *b*) and from node *b* to node *a*, the link is said to be *symmetric*. If communication is possible in only one direction, the link is said to be *asymmetric*. Uncertainties in radio propagation (e.g. atmospheric interference or physical obstacles) may make links between otherwise identical nodes with identical, omni-directional antennas, asymmetric.

A node *c* is further said to be a *two-hop neighbor* of the node *a* if node *a* and node *c* are not the same node and if node *c* is not a neighbor of node *a* and if there exists a symmetric link between node *c* and a node in the neighborhood of node *a*, with which node *a* has a symmetric link.

2.1.2. Neighbor Sensing in OLSR

OLSR aims at being completely independent of the underlying link-layer being used. Thus, even though OLSR might utilize information available from an underlying link-layer, characterizing the existence and quality of the links, the protocol specifies mechanisms for neighbor sensing, including the ability to detect the link "characteristics" (in terms of symmetry or asymmetry as described above).

In OLSR, a node emits HELLO-messages periodically. Changes in the neighborhood is detected from the information in these messages. A HELLO-message contains the emitting node's own address and the list of neighbors known to the node, including the status of the link to each neighbor (e.g. symmetric or asymmetric). A node thereby informs its neighbors with which neighbors, and in what direction(s), communication has been confirmed.

Upon receiving a HELLO-message, a node can thus gather information describing its neighborhood and two-hop neighborhood, as well as detect the "quality" of the links in its neighborhood: the link from a node *a* to a neighbor *b* is symmetric if in the HELLO-message from *b* the node *a* sees its own address (with any link status) - otherwise the link is asymmetric.

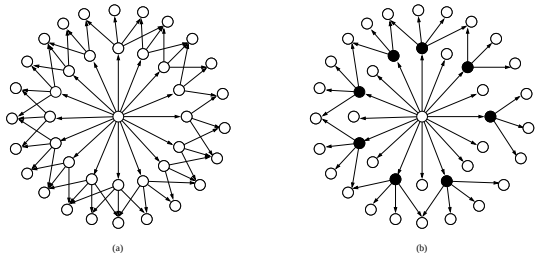

Figure 1: *Example of pure flooding (a) and flooding using Multipoint Relays (b). The source of the message is the node in the center. Each arrow pointing to a node, indicates that a node receives a copy of the message. The filled nodes are selected by the center node as Multipoint Relay.*

Each node maintains an information set, describing the neighbors and the two-hop neighbors. Such information is considered valid for a limited period of time, and must be refreshed periodically to remain valid. Expired information is purged from the neighbor- and two-hop neighbor sets.

2.2. Generic message flooding

HELLO-messages are exchanged between neighbors only, and provide a node with topological information describing its neighborhood and two-hop neighborhood. However, since MANETs can be of arbitrary size, a method for flooding topological information into a network of any size is required. Thus OLSR introduces an efficient, generic way of flooding control traffic to all nodes in the network.

The task at hand is to provide a method for a node to emit a control message, which will be flooded into the entire network in an "efficient" way. In this context, we define "efficient" to mean that the interface of each node receives the same, flooded message once. This implies, that all nodes actually receive the message, as well as that there have been no unnecessary, duplicate retransmissions of the message.

The message flooding mechanism in OLSR is based on an optimization of the simple flooding strategy, where all nodes receiving a message to be flooded, relay (forward) the message the first time they receive the message. Thus, a node would perform some "local duplicate elimination", while there would be nothing preventing two neighboring nodes from relaying the same message to the same nodes, as illustrated in figure 1a.

Ignoring message loss due to collisions, this ensures that all reachable nodes receive a message at least once - though, as illustrated in figure 1a, quite likely more than once. This is a problem since, when a message is transmitted over the wireless medium, all other nodes within radio range of the transmitting node will either have to remain silent, or will cause message loss due to collisions.

In OLSR, the problem of duplicate transmissions of a message within a region is addressed through the notion of multipoint relays (MPRs)[5]. A node selects a set of symmetric neighbor nodes as MPRs, implying that each node has a (possibly empty) set of nodes, which have selected it as multipoint relays. This set is called the MPR selector set. A node, then, has the responsibility of relaying messages, emitted from its MPR selectors, while not relaying messages from any other nodes. As illustrated in figure 1b, careful selection of MPRs (the filled nodes) may greatly reduce duplicate retransmissions.

While selecting MPRs, a node utilizes information describ-

ing the two-hop neighbors, as acquired from the neighbor sensing process. All nodes select their MPRs independently, possibly choosing different algorithms / heuristics for selecting a "minimal" MPR set. The invariant for the algorithms being, that a message, emitted by the node and relayed by its MPRs, must reach all the node's two-hop neighbors.

A node is informed of its MPR selector set through information, piggybacked to the HELLO-messages.

In order for a node to forward a control message, the following conditions must thus be satisfied:

1. the message must be meant to be forwarded (indicated by information in the header of the message),
2. the message must not have been received by the node before,
3. the node must have been selected as MPR by the node, from which the message was received

The OLSR protocol specification [1] defines a generic message format and an algorithm for processing such messages. This includes time-to-live considerations etc., out of scope for this description.

2.3. Topology information

With mechanisms for neighbor sensing and for flooding messages to all nodes in the network in place, the final task for the routing algorithm is to diffuse a sufficient set of topological information to all nodes in the network.

In OLSR, all nodes with a non-empty MPR selector set periodically generate a topology control message (TC-message). This TC-message is diffused to all nodes in the network as described in section 2.2. A TC-message contains the address of the node generating the TC-message, as well as the addresses of all the MPR selectors of that node. Thus through a TC-message, a node effectively announces reachability to all its MPR selectors. Since all nodes have selected an MPR set, reachability to all nodes will be announced through the network. The result is that all nodes will receive a partial topology graph of the network, made up by all reachable nodes in the network and the set of links between a node and its MPR selectors. Using this partial topology graph, it is possible to apply a shortest path algorithm for computing optimal routes from a node to any reachable destination in the network [1].

The topological information in each node is valid for a limited period of time, and must be refreshed periodically to remain valid. Expired information is purged from the topology graph.

3. Experimental results

Our practical experiments were conducted on a test network of desktop and laptops, equipped with IEEE 802.11 wireless interfaces. Our test setup spanned from 3 to 10 nodes, and a number of different experiments were conducted. The purpose of the experiments were to serve partially as a "proof of concept", partially as a way to indicate areas for possible optimizations and further investigations through simulations of larger setups.

3.1. Proof of concept

We first set up our test network as illustrated in figure 2a, and observed, that routing was indeed established from node *a* to node *f*, following the path indicated by the arrows. Using node *a* as a traffic source[1] and node *f* as a destination, we removed node

[1] Effectively, node *a* was forwarding a RealAudio stream

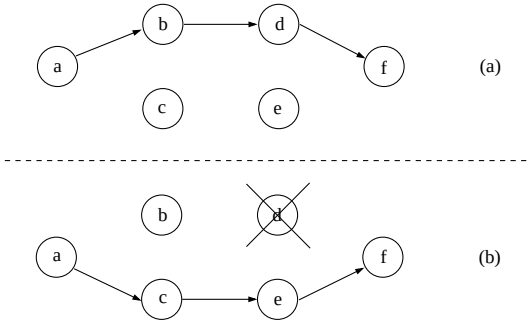

Figure 2: *Example of routing in OLSR. Node a is a traffic source, node f destination for traffic. (a) indicates the used path from a to f. (b) indicates the path after the routing protocol was able to recover after node d crashes (indicated by the cross-out of node d).*

d from the network, as illustrated by the "cross" in figure 2a. We observed, that after a short time (bound by the time neighbor information is stored in each node), a new route was established, following the path indicated by the arrows in figure 2b.

3.2. Observations

Through further experiments with OLSR, the following interesting observations were made:

1. Occasionally, in an otherwise static setup, routes to part of the network would temporarily disappear, only to reappear moments later. Investigations into this matter, revealed that this was due to the fact that the periodic emission of control messages from the individual nodes had become synchronized. I.e. neighboring nodes would, at the same time, attempt to transmit a TC-message, causing both messages to be lost due to collision. With the same interval between the periodic transmissions in all nodes, a number of sequential TC messages would be lost, causing topological information to expire in the network. Experiments indicated that enforcing jitter (a small, random delay) on sending each control messages resulted in fewer control messages to collide, and hence in more stable routes.
2. In the same situation as above, it was found that piggybacking several control messages (e.g. TC messages to be forwarded from different other nodes, as well as a locally generated TC and HELLO message) eased the load on the network. This was due to the fact that through piggybacking messages, fewer medium access cycles were required for control traffic. Piggybacking was introduced by allowing the generic message forwarding mechanism to introduce a small delay before forwarding a message.
3. Occasionally, a node would, based on exchange of a single, transient pair of HELLO messages, detect another node as a symmetric neighbor. Despite that the two nodes only occasionally could communicate directly (i.e. the link was not stable in the symmetric state), OLSR would choose to route traffic to that node directly, rather than through another, intermediate, node with stable links. It was found, that strengthening the requirement for accepting a node as a symmetric neighbor (e.g. to require that a number of HELLO-messages are received within a certain time-frame for a link to be "up")

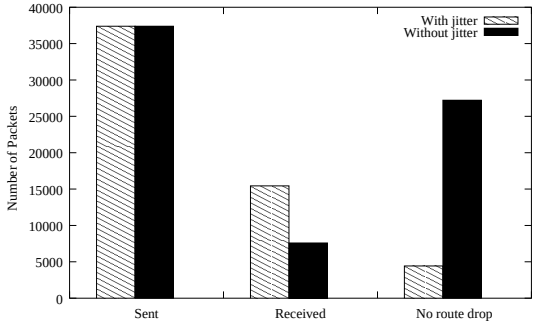

Figure 3: *The average number of packets sent, received, and dropped because of route unavailability. The number are averages from test sets with and without enforced jitter on transmission of control packets.*

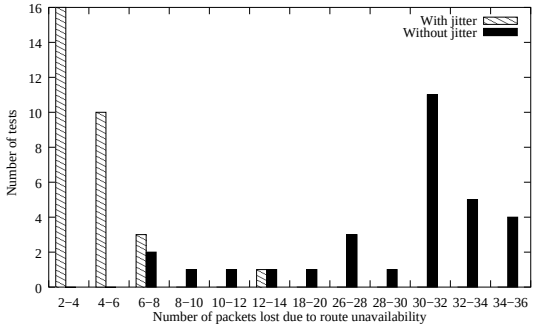

Figure 4: *The number of tests as a function of number of packets lost due to route unavailability. The number of packets are showed in intervals of thousands.*

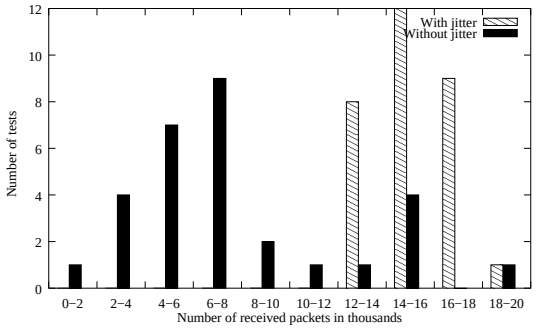

Figure 5: *The number of tests as a function of number of packets received. The number of packets are showed in intervals of thousands.*

made the network more resistant against such transient situations - at the expense of a slower response to new links.

4. Simulations

A distinguishing feature of the simulations presented in this paper is the fact that the simulated scenarios are neither completely "random" nor selected to expose specific aspects of the protocol. Rather, an automatic scenario generator has been developed. This scenario generator can take a set of simulation parameters (simulation time, mobility, number of nodes, communication parameters, field simulation size, groups) and generate a number of random scenarios, which may be different, but yet conforming to the parameters of the situation. This enables automatic simulation of a large number of scenarios with the same characteristics. Thus the effects of a favorable pick on the simulation results is averaged out.

Our simulations are performed using the ns2 [6] network simulator. A test set with 30 simulations are performed and statistical tools have been used to analyze the results. The simulations are all generated with the same standard parameters. The field size is 1000x1000 meters with 50 nodes moving from 1 to 5 m/s and resting between 0 and 6 seconds at each way-point. The simulation time is 250 seconds with 25 traffic streams of 64 bytes packets send with an 0.10 second interval.

4.1. Enforced Jitter

To test the effect of enforced jitter on the transmission of control messages, test sets with and without jitter were performed. The jitter was chosen from the interval $[-0.5; 0.5]$.

Figure 3 shows the average amount of packet that were sent, received, and dropped due to route unavailability. As the figure shows the same amount of packets were sent in both test sets. The number of received packets, and packets dropped because of no route are however very different. Without jitter half as many packets reaches their destination as with jitter, while the amount of packets lost due to route unavailability is six times the amount of the simulations with jitter.

Statistical analysis furthermore showed that the dispersion of the results without jitter was substantially larger than those with jitter as seen in figures 4 and 5. This means that the performance with enforced jitter is not only better, but also more stable than without jitter.

4.2. Piggybacking

When testing the effect of piggybacking control messages, jitter was initially enforced due to the good results as stated above. Test sets with holdback time (i.e. the maximum time a message was held back, waiting for other messages to be piggybacked onto) of 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0 seconds were performed.

Figure 6 shows the average amount of packet that were sent, received, and dropped due to route unavailability in the situation where the nodes enforced both jitter and piggybacking. As the figure shows the same amount of packets were sent in all test sets. No significant impact of a shorter or longer holdback time can be observed. An initial hypothesis was, that the effects of piggybacking were, in fact, masked out by the jitter - which, as described, could offset the emission by up to $\pm$ 0.5s and thereby implicitly facilitate piggybacking.

Thus, to confirm this, we conducted the same simulations without jitter, as illustrated in figure 7.

It can be observed, when comparing to figure 6, that the effect of jitter is significant and that, without jitter, substantially more packets are dropped due to a route to the destination being unavailable.

In a network without mobility - and hence without link breakages - OLSR with piggybacking performed as illustrated in figure 8: the ratio of received packets is much higher than in the situations with mobility.

In figure 9, we compare running OLSR with and without piggybacking and with and without jitter. Figure 9 show the average number of packet from the test sets without anything, with enforced jitter, with jitter and piggybacking (holdback time: 0.2

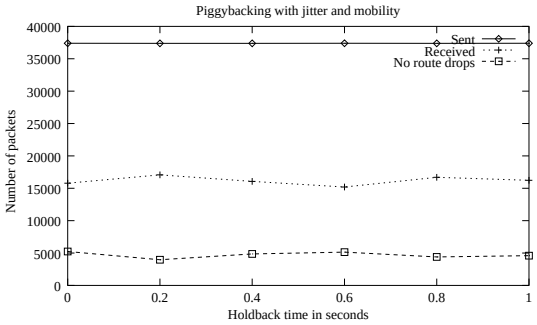

Figure 6: *The average number of packets sent, received, and dropped because of route unavailability, with both jitter and piggybacking in place. The numbers are averages from test sets with holdback time from 0.0 to 1.0 seconds*

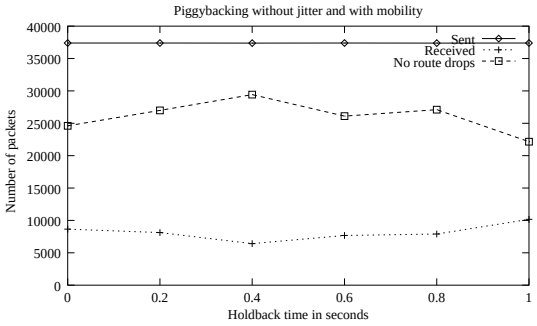

Figure 7: *The average number of packets sent, received, and dropped because of route unavailability with piggybacking only. The numbers are averages from test sets with holdback time from 0.0 to 1.0 seconds.*

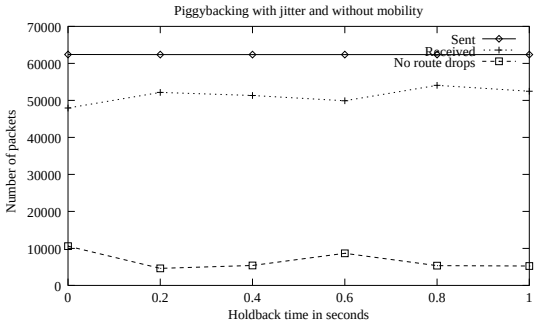

Figure 8: *The average number of packets sent, received, and dropped because of route unavailability using piggybacking in a network without mobility. The number are averages from test sets with holdback time from 0.0 to 1.0 seconds.*

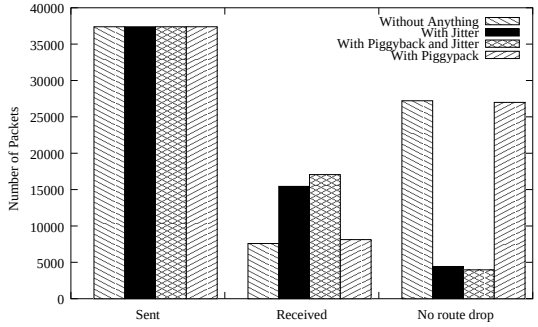

Figure 9: *The average number of packets sent, received, and dropped because of route unavailability. The number are averages from test sets of "pure" OLSR, OLSR with jitter, OLSR with piggybacking and OLSR with both piggybacking and jitter.*

seconds), and finally without jitter but with piggybacking (holdback time: 0.2 seconds). We have chosen to show the results with piggybacking with a holdback time of 0.2 seconds, because this gave the best results in the piggyback tests. The figure shows, that the effects of piggybacking are not significant, while jitter has a high, positive impact. While not an impossible situation, this does contradict our experiments, in which both piggybacking and jitter was found to have a positive effect on performance. We will return to this issue in section 5.

5. Discussion

It is our belief, that through using a scenario generator and conducting a large number of simulations, the results presented are more fair and trustworthy than if a favorable pick of a few scenarios had been performed to exhibit the protocols positive or negative sides. That said, there are evidently discrepancies between the experimental results and the data obtained by simulations. While the introduction of jitter yielded positive results in both the experiments and in the simulations, the introduction of piggybacking in the simulations had virtually no effect in contrast to observations from the experiments. The course of this discrepancy is currently unidentified. We do, however, suspect that it may be due to either the "perfect" transmission model of ns2 (e.g. no interference from the environment) as well as discrepancies between the actual link layer used, and the one being simulated by ns2. This is an area where further investigations are required.

The improvements to the neighbor sensing mechanism, suggested by the experiments, for identifying only "stable" neighbors as symmetric (and thereby allowing those to be used for data traffic), have not been simulated. While simulations and further testing is desirable, it is unclear how much simulations would reveal: the transmission model of ns2 is perfect. The common real-life situation, where transient reflections or drop in background noise would allow only some of the HELLO-messages to be exchanged, will not be captured by the simulations. Realistic simulations and exhaustive testing hereof is another area where further investigations are required.

An issue, which has not been discussed through this paper, is the optimality of the proposed optimizations. E.g. for piggybacking, what is the optimal "deadline" at which all queued messages in a node must be sent? The simulations and the experiments have been performed using a constant holdback time, however other metrics could be envisioned. A node could, e.g., choose to hold all messages it is required to forward until it itself is due to emit a control message (HELLO or TC). Alternatively, using a constant holdback time, a node could force emission of extra HELLO- and/or TC-messages, to increase the networks reactiveness to topology changes.

Likewise, different algorithms for detecting a neighbor node can be proposed. In our experiments, we required that a constant ratio of HELLO messages were delivered (e.g. 2 out of 3 expected). Alternative proposals could include various

metrics for weighting HELLO messages, giving more weight to recent messages while still taking the older history of the link into account when estimating stability.

6. Conclusions

Our experiments and simulations have shown that the proposed protocol for routing in MANETs works - in most situations, it even works well. Our experiments revealed shortcomings in a couple of situations. Noticeably, the neighbor sensing mechanism did, occasionally, cause a transient (and hence unstable) link to be reported as "symmetric", and thus eligible for routing. Also noticeably, loss of control messages due to collisions on the link layer caused temporary loss of routes to some destinations in the network.

We suggested and implemented solutions to these problems, in form of jitter, piggybacking and improvement to the neighbor sensing mechanism, and observed that they did provide significant improvements. Simulating jitter and piggybacking in ns2 qualified the statement that using jitter with OLSR was a significant improvement over OLSR alone, while no significant effect could be determined from piggybacking. The course for the discrepancy between the experiments and the simulations on the topic of piggybacking is to be determined, however is suspected to be due to differences between the link layer used for the experiments and the link layer used in the simulations.

7. Acknowledgments

The authors would like to thank Philippe Jacquet, Project Hipercom, INRIA Rocquencourt, for providing access to test facilities and for supporting the intellectual efforts behind this paper.

The authors would also like to thank Nicolai Larsen and Elena Tørnæs Beck for commenting on this paper in its final stages.

8. References

[1] Philippe Jacquet, Paul Muhlethaler, Amir Qayyum, Anis Laouiti, Laurent Viennot and Thomas Heide Clausen "Optimized Link-State Routing Protocol", draft-ieft-olsr-04.txt - work in progress, March 2001.

[2] Charles E. Perkins, Elizabeth M. Royer and Samir R. Das "Ad hoc On-Demand Distance Vector (AODV) Routing", draft-ieft-aodv-08.txt - work in progress, March 2001.

[3] David B. Johnson, David A. Maltz, Yih-Chun Hu, Jorjeta G. Jetcheva "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks", draft-ieft-dsr-05.txt - work in progress, March 2001.

[4] Kevin Grace "Mobile Mesh Routing Protocol", draft-grace-manet-mmrp-00.txt - work in progress, September 2000.

[5] Amir Qayyum, Laurent Viennot and Anis Laouiti "Multipoint relaying: An efficient technique for flooding in mobile wireless networks", INRIA research report RR-3898, 2000.

[6] Network Simulator - ns - 2, Available at http://www.isi.edu/nsnam/ns/