Analyzing the Impact of Neighbor Sensing on the Performance of the OLSR protocol

Michael Voorhaen and Chris Blondia

University of Antwerp, Dept. Mathematics and Computer Science
Middelheimlaan 1, B-2020 Antwerpen, Belgium
{firstname.lastname}@ua.ac.be

***Abstract*— This paper presents an analysis of several neighbor sensing approaches for the OLSR routing protocol. While several performance studies of OLSR proceed this work, few attention has been paid to the impact of neighbor sensing on the performance of ad hoc routing protocols. The goal of this article is to better understand how neighbor sensing can contribute to packet loss in an OLSR network and thus degrade the overall performance. Three neighbor sensing schemes are compared: the OLSR HELLO messaging protocol, Fast-OLSR and a link-layer feedback scheme that uses information of the 802.11 MAC to determine lost links. To allow more detailed analysis of the events occuring in the network we initially limit our simulation setup to a simple scenario. As a result we are able to seperate the loss of packets into loss due to the neighbor sensing mechanism used and loss due to the impact of neighbor sensing on the other protocol operations. In the second part of this paper we compare the performance of the link-layer feedback scheme to OLSR in a random waypoint scenario.**

I. Introduction

Mobile Ad hoc NETworks (MANETs) are an emerging technology made possible by the progress in the field of wireless communication. Target applications for these networks can be envisaged for almost every aspect of modern day life, ranging from home automation systems, electronic learning environments, e-health, vehicular networks up to public safety communication systems. In each of these applications ad hoc networks are faced with dynamic environments and neighboring nodes can be discovered or lost at any moment, causing the network topology to change constantly. Ad hoc routing protocols can be classified into proactive, reactive and hybrid routing protocols. Proactive protocols are very similar to routing protocols from wired networks, since they regularly advertise routing information into the network and each node keeps a route to each other node in the network (e.g. OLSR [6]), however support for mobility in the network has been added. Reactive protocols follow the reasoning that in an ad hoc network a route to a node is not needed until it is actually used. Therefor routes are determined on-demand, often by a so called route request packet that is broadcasted into the network (e.g. AODV [7], DSR [8]). Hybrid protocols combine the benefits of both protocols by working proactive in the local neighborhood of a node and reactive for nodes further away. All of these protocols have one common part, they need to be able to know what nodes they can reach directly. In this paper we will refer to this as neighbor sensing. The primary purpose of neighbor sensing is to discover information on the local topology, i.e. discovering new neighbors and timely discovery of losing a direct link with a neighbor. The latter is critical if packet loss is to be avoided. The intention of this paper is to investigate the impact of neighbor sensing on the performance of the OLSR routing protocol [6]. We choose the OLSR routing protocol because it clearly defines its own neighbor sensing protocol in the RFC, and several alternatives to this protocol have already been proposed.

A. OLSR

The OLSR (Optimized Link State Routing) protocol is a *proactive* ad hoc routing protocol. Its operation is similar to classic link state routing protocols, however to avoid the overhead, inherent to advertising the link state information, a clever flooding optimization is used. The OLSR protocol consists out of four major elements:

- *Neighbor Sensing:* Each OLSR node gathers information about its local neighborhood from the *HELLO messages* that it receives. The operation of the HELLO messaging is explained in detail in section II-A.
- *MPR Selection:* From its neighbor set OLSR chooses a small amount of *Multi Point Relays* (MPRs) that will be used to optimize flooding of routing signalling packets.

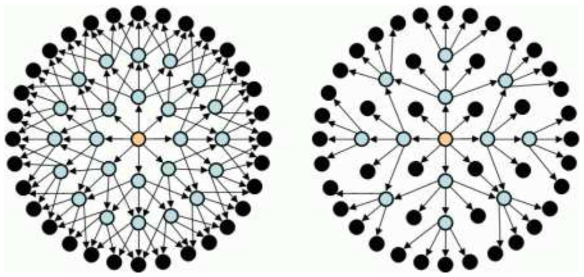

Fig. 1. Optimized flooding in OLSR

- *Optimized Flooding:* To decrease the overhead of advertising link state information only nodes chosen as MPR send out the Topology Control (TC) messages. The *TC messages* that MPRs broadcast only contain the links of their MPR selectors, to limit the packet size. This approach reduces the signalling overhead, while still

guaranteeing that there exists a route between each pair of nodes that can reach each other.
- *Route Selection:* Routes are computed using a link state algorithm that is similar to the Dijkstra algorithm.

B. Related Work & Outline

[12] analyzes the performance of the MPR selection algorithm used in the OLSR protocol. [11] points out some problems in the packet processing of several OLSR implementations that impacts the convergence of the MPR selection algorithm towards a stable set of MPRs after a new link is discovered. Section II describes the OLSR neighbor sensing protocol, the fast-OLSR extension, and an approach that enhances OLSR neighbor sensing using feedback from the 802.11 link-layer. The simulation study that we performed is described in sections III and IV. Section V concludes this paper and proposes some future research perspectives.

II. Neighbor Sensing

A. OLSR Neighbor Sensing

The OLSR RFC [6] proposes a HELLO messaging protocol to perform neighbor sensing. In order for a node to discover its neighbors, HELLO packets are broadcasted periodically. The symmetric nature of a link is determined by advertising the neighbors from which a node has received a HELLO message in its own HELLO messages. We will first introduce some additional notations to express the neighbor set and the contents of the HELLO messages. The neigbor set of a node N_x that has N_s as a symmetric neighbor and N_a as an assymetric neighbor can be written down as a tuple of the assymetric and the symmetric links: $NS_x = (A_x = \{N_a\}, S_x = \{N_s\})$. The following expression represents the contents of a HELLO message from N_x: $HELLO_x = (N_x, A_x, S_x)$. The operation of the HELLO messaging protocol between two nodes, called N_1 and N_2 can then be explained as follows:

- Both N_1 and N_2 start out with empty neighbor sets: $NS_1 = (\emptyset, \emptyset)$ and $NS2 = (\emptyset, \emptyset)$.
- N_1 sends out a HELLO message which is received by N_2. Since N_1 does not know any neighbors the neighbor list in the HELLO message is empty: $(N_1, \emptyset, \emptyset)$.
- On receiving this message N_2 adds N_1 to its neighbor set and marks the link to N_1 as asymmetric. The update neighbor set of N_2 is then $NS_2 = (\{N_1\}, \emptyset)$.
- N_2 broadcasts a hello message containing N_1 in the list of asymmetric neighbors $(N_2, \{N_1\}, \emptyset)$.
- N_1 receives the hello message and adds N_2 to its neighbor set. Since N_2 advertised an asymmetric link to N_1, the latter realizes that N_2 is receiving its HELLO messages. Thus it concludes that a symmetric link with N_2 exists. This results in the following updated neighbor set of N_1: $(\emptyset, \{N_2\})$.
- N_1 broadcasts a hello message and advertises N_2 as a symmetric neighbor: $(N_1, \emptyset, \{N_2\})$.
- On receiving this message N_2 realizes that it has a symmetric link to N_1: $NS_2 = (\emptyset, \{N_1\})$.
- Periodic broadcasting of the HELLO messages is used to keep the entry of the link between N_1 and N_2 alive, a link is only considered lost if its entry times out before another HELLO message was received.

In addition to discovering new neighbors the OLSR HELLO messaging protocol is used to determine if links are symmetric. OLSR only takes into account symmetric links in its route computation algorithm. To optimize the flooding of control messages, an OLSR node also chooses some of its neighbors to act as a Multipoint Relay (MPR). The nodes chosen as MPR are also advertised in the HELLO messages. These entries can also be considered as symmetric links since only nodes to which a symmetric link exist can be chosen as MPR. Additionally a node can build up a view of its two hop neighborhood using the information gathered from the HELLO messages it receives.

B. Link-layer Feedback extension

We now propose a lightweight link-layer feedback extension to the OLSR protocol that works alongside the HELLO messaging protocol and attempts to improve the reaction time for detecting a link down event.
Figure 2 shows how a simple link-layer feedback scheme, that reacts to link-layer failures, can be implemented. This figure shows a rough description of how a router works internally. Packets arriving from the network are checked, to see if they have reached their destination, if so they are delivered to the system. Packets from the system or those that have not yet reached their destination will be forwarded and a route lookup is performed. If no route is found the packet is discarded, else the packet is delivered to the link-layer (in our case 802.11). Packets that 802.11 can not deliver to the next hop are sent back up the network stack to the network layer, i.e. OLSR (LL Feedback). In order for OLSR to determine what the next hop was it needs to perform a reverse ARP lookup in the ARP cache. If this lookup succeeds the link entry in the OLSR Link Table is marked as asymmetric and the routing table is updated. OLSR will then attempt to reroute the packet through another node. Marking the link as assymetric makes sure that the next time OLSR sees a HELLO message from this neighbor it will reactivate the link and it can be used again, making sure that the three-way handshake that is used normally is no longer necessary.
It is important to remark that in the case that 802.11 does not provide correct information (e.g. the network is too congested to send the packet), OLSR will also mark the link as assymetric. The impact on the protocol should be minimal since it only requires one HELLO packet to recover and the same situation where one node has marked the link as symmetric and the other node has not can also occur when the three-way handshake is not yet completed.
In our simulations we set the short retry limit, i.e. the number of times 802.11 tries to send the packet, to the default value specified by the protocol: 7. This means that 802.11 will attempt to send a packet 7 times before assuming that it cannot reach the next hop.

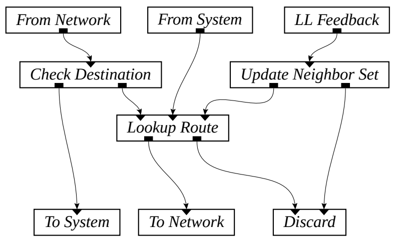

Fig. 2. Link-layer Feedback Extension

Parameter	**value**
HELLO_INTERVAL	$1s$
TC_INTERVAL	$5s$
NEIGHB_HOLD_TIME	$3s$
TOP_HOLD_TIME	$15s$

Fig. 4. Default OLSR Parameters

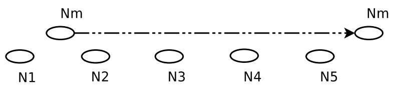

Fig. 3. Strip Scenario

Parameter	**value**
Node coverage	$100m$
802.11 Rate	$2Mb/s$
Application type	CBR
Packet size	$500byte$
Rate	$64kb/s$

Fig. 5. Simulation Parameters

C. *Fast-OLSR*

Fast-OLSR [9] assumes that in a highly mobile ad hoc network the default HELLO frequency is not enough to track the motion of the node. It attempts to improve upon the OLSR neighbor sensing protocol by introducing additional signalling when the mobility increases. A node switches to Fast-OLSR when it notices that it is moving fast (e.g. by keeping track of the changes in its neighborhood). When in Fast-OLSR mode the node starts broadcasting fast-HELLOs, which are generated at a higher frequency then the default HELLO messages. To reduce the overhead caused by the increased frequency of the fast-HELLOs, they contain a limited number of neighbor addresses. The nodes receiving a Fast-HELLO will then reply with a fast-HELLO, and from the nodes that replied a small set of MPRs is chosen. In this way a set of Fast-Links are established which are kept alive using fast-HELLOs. If an MPR misses a fast-HELLO from the moving node it broadcasts a TC message declaring the lost link. In this way loss of a link can be detected faster than using the default OLSR neighbor sensing protocol.

III. Strip Scenario

In the following sections we will investigate in detail the benefits and problems related to these neighbor sensing approaches. We take a look at a simple strip scenario, shown in Fig. 3. In real life this could compare to a highway scenario where one car is driving faster than the other cars, or a variety of other examples. For our simulations we will use the nsclick [2] simulation platform. Nsclick embeds the Click Modular Router Platform ([3]) into the ns-2 [1] simulator allowing us to run actual routing protocols developed for the click platform inside a simulation. We based our work on a click implementation of the OLSR protocol . The parameters used for OLSR are shown in Fig. 4.

In our simple scenario only one application is running, which is a CBR stream between the first node and the moving node. Different speeds for N_m where simulated: from $2.0m/s$ to $20.0m/s$ in 10 steps. A summary of the important parameters can be found in Fig. 5.

A. *Results*

Figures 6 and 7 show the packet receive (Rx) and drop events for the OLSR neighbor sensing protocol and OLSR with the link-layer feedback extension respectively. We will start by discussing the results for a standard OLSR implementation.

a) OLSR Neighbor Sensing: Figure 6 shows that each time the destination moves out of range of the last hop several packets are lost. In total four gaps can be found in the packet arrival graph, corresponding to N_m losing its direct connnection with N_1, N_2, N_3 and N_4 respectively. This loss is caused by the HELLO messaging protocol of OLSR. As specified in table 4 the HELLO interval was set at the default value for OLSR, which is 1 second, and the HELLO timeout value is set to 3 seconds. This means that until the neighbor entry of N_m times out, OLSR will keep routing the packets to the destination while it is no longer in range.
As can be seen in Fig. 11 no trend in the packet loss can be observed with different speeds of N_m.
The effect of the HELLO interval on the packet loss is shown in Fig. 9. We can conclude that even if we drastically decrease the HELLO interval there is still packet loss when the data rate is high. In fact the rate of the HELLO messages should be a function of the rate at which a node is forwarding packets and not the mobility, since the latter only affects the rate at which the link down events occur. Fig. 10 shows that the amount of routing signalling packets increases exponentially when the HELLO interval decreases linearly, as was to be expected.

b) Link Layer Feedback: Figure 7 shows the benefit of the link-layer feedback mechanism, however there are still two gaps with packet loss. We will now provide a detailed analysis of the events occuring in this simulation.
No packets are lost when the link between N_1 and N_m disappears, since N_1 can immediately reroute the packets through N_2, when it notices the lost link. The same argument

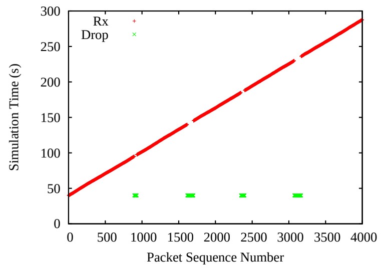

Fig. 6. Pakket arrival events for OLSR (Packets dropped are shown at 40.0s)

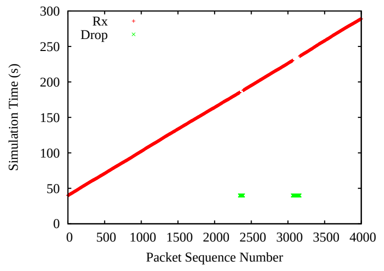

Fig. 7. Pakket arrival events for OLSR-LL (Packets dropped are shown at 40.0s)

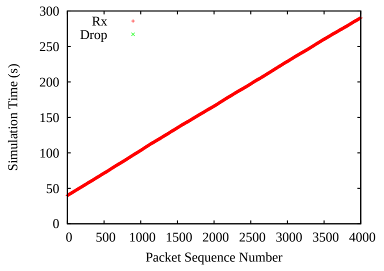

Fig. 8. Pakket arrival events for OLSR-LL with additional MPRs (Packets dropped are shown at 40.0s)

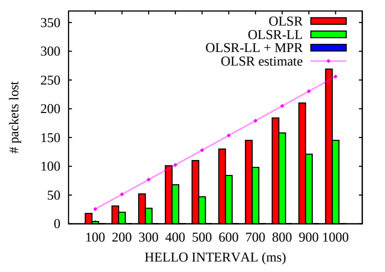

Fig. 9. # packets lost for different values of HELLO_INTERVAL

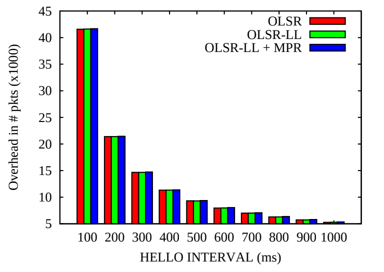

Fig. 10. Routing Signalling Overhead

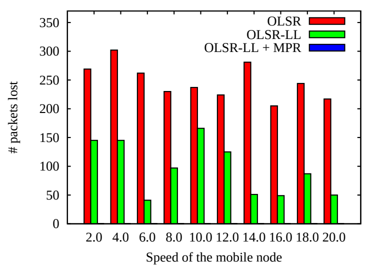

Fig. 11. # packets lost with varying node mobility

holds for the link between N_2 and N_m.
Before N_m loses connection with N_3 it will have connectivity with N_3, N_4 and N_5. This is shown in Fig. 12. Before the link with N_5 existed, N_m needed to choose N_4 as MPR since it was the only neighbor through which it could reach N_5. At the moment N_m discovers that it has a symmetric link to N_5 it will no longer be necessary to choose N_4 as an MPR, causing N_4 to stop advertising its link to N_m.
On the exact moment that N_3 loses its link to N_m, there will be no more nodes in the network to advertise that they have an existing link to N_m. While N_3 learns of the lost link from the link-layer feedback, the only way for N_m to notice this is when the neighbor entry times out. In this case the packet loss does not occur at N_3 but at N_1 and N_2 since they will not have a route to N_m once they learn of the broken link between N_3 and N_m. This also means that it will take $NEIGHB\_HOLD\_TIME$ for N_m to determine that the link to N_3 - its only MPR - has been lost, and to choose a new MPR (i.e. N_4). After which N_4 still needs to send a TC packet before the rest of the network is informed. We can follow the same reasoning for the packet loss occuring when the link with N_4 is lost, since there is no reason to choose N_5 as MPR.
Note that this packet loss is an edge effect. If N_4 and N_5 had not been the two last nodes in the chain we would not have lost any packets. Although this scenario can only happen with edge nodes that only have 1 MPR, there are two reasons we are discussing it here:

- It is partially caused by the neighbor sensing.
- While the information about the lost link that N_3 advertises into the network using a TC message is up-to-date, the lack of information on other existing links to N_m is not.

The asymmetric behaviour of the neighbor sensing, by which we mean that the sending node has discovered the lost link, but the receiving node takes much longer to discover this, is at the bottom of this problem.

c) MPR Coverage: One possible solution is to make sure that more topology information is available in the network. This can be done in a number of different ways, but we have chosen for the solution proposed in [10]. The authors propose a tunable parameter for the OLSR protocol called *MPR coverage*. If MPR coverage is set to 1, a minimal MPR-set, minizing both overhead and redundancy, is sought. With MPR coverage set to k, a node will choose at least k MPRs, meaning increased overhead and redundancy. We implemented this solution and ran the same simulations as before, only setting MPR coverage to 2, forcing the OLSR nodes to choose at least 2 MPRs. Figure 8 shows the packet arrival events for this simulation and confirms that no packets are lost. In combination with the link-layer feedback this solution succeeds in avoiding any packet loss in the simple scenario we set up. Figures 11 and 9 show that these results hold for different node speeds and data rates, i.e. no packet loss are lost. Figure 10 shows that setting the MPR coverage to 2 does not have a significant impact on the overhead in the strip scenario.

d) Fast-OLSR: Fast-HELLO messages are not that different from HELLO messages: they are sent at a higher rate and contain information about less neighbors. This means that the main benefit of Fast-OLSR is reducing the overhead by sending smaller HELLO messages, and not by sending less messages. In [9] the fast-HELLO_INTERVAL is chosen to be 100ms instead of 1s. However our results show that the amount of packets lost is not a function of the speed of the node, but of the rate at which it is generating traffic. Secondly packets can still get lost when a node has not entered fast-OLSR mode.

IV. Random Waypoint Scenario

We have taken a look at the strip scenario since it was easy to understand and analyze. Even though the results were quite predictable, the scenario was important in helping us understand the finer protocol operations of OLSR. Additionally we simulated a random waypoint scenario with 40 nodes on a 1000m by 1000m surface, 250m transmission range and 20 CBR streams with different speeds for the mobile nodes: $2.0m/s$, $4.0m/s$, $6.0m/s$, $12.0m/s$, $20.0m/s$ and for each node speed 3 different rates for the 20 CBR streams were simulated: $16kb/s$, $32kb/s$ and $64kb/s$. The same settings for OLSR were used and the link rate was increased from $2Mb/s$ in the strip scenario to $54Mb/s$.

Due to the complexity and size of the traces it was not possible to perform a complete analysis of the obtained results. Future work will be to translate the experience from small scenarios like the strip scenario into tools that allow us to perform a detailed analysis of these random waypoint scenarios. This section presents an overview of the analysis performed so far and is meant to validate the results obtained in section III.

A. Results

Figure 13 shows the packet delivery ratio for the different node speeds and data rate. The results show that our simple link-layer feedback approach increases the performance when the network load is acceptable, at a high network load the link-layer feedback has a negative impact on the network performance. The effect of the link-layer feedback scheme can especially be noticed when the speed is higher and the network load is low. Indeed at higher speeds this approach increases the packet delivery ratio by up to 30%. As can be seen in Fig. 13 the performance of the link-layer sensing decreases with a higher network load for the different speeds. This is because the link-layer feedback of 802.11 becomes less accurate at a high network load, i.e. failure of delivering a packet to the next hop could also mean that the medium was too congested and the node was not able to gain access to the medium.

We also measured the probability of receiving wrong feedback from the link-layer, by comparing the time at which an event was received with detailed topology information obtained by using the analysis tools provided by the BonnMotion software. The results are plotted in Fig. 15, that shows that the amount of false positives depends on the occupancy of the medium when the speed is low. When the speed is high links will usually not live very long so the probability of detecting a broken link while it is still there is a lot smaller. When the network load is low there is less probability of having a congested link-layer, which means that there is less probability of having wrong feedback. However at higher speeds the link lifetime reduces and the probability of having false positives is higher. It appears that the accuracy of the link-layer feedback scheme can be improved, however the false positives only indicate that the performance can still be improved. We also observed (not shown in the graphs) that the number of L2 events received is much larger that the actual number of link breaks in the scenario, which was unexpected. By carefully observing the traces we noticed that often several packets are sent up from the link-layer to the network layer that point out the same broken link, this occurs more frequent in scenarios with a higher network load.

Figure 14 confirms these conclusions: the average number of packets buffered at the link-layer in the high load scenario is around 30 packets. If we have a look at the maximum occupancy together with the average number of packets that are lost due to buffer overflows, we observe that when half of the applications have been started, at any point there is at least one node that has a full buffer and on average between 20 and and 40 packets are lost each second. This implies that when a link between two nodes dissapears there can be several packets residing in the link-layer queue. In our simple scheme none of these packets can be delivered to the next hop and eventually they will sent back up the network stack when they can not be delivered. Since packets to other neighbors will also be residing in this queue, it might take longer to detect the link break since we only notice a possible lost link when attemping to send the packet. In an ideal scenario a packet would be served immediately by the link-layer and the following packets would follow another route. This means that there is an effect of the network load on the accuracy and the behaviour of the link-layer feedback scheme.

Our results also imply that with an increasing load the overall network performance could be increased if the MAC layer does not only return the packet - whose transmission failed - to the network layer, but also all the other packets remaining in the link-layer queue with the same next hop. This would reduce the amount of unnecessary transmissions and unnecessary buffer usage. We did not yet investigate the possible impact of this, however this can have a significant impact on the complexity of the system.

V. Conclusion

In this paper we have presented an analysis of several link sensing approaches for the OLSR protocol. Using the nsclick simulation environment we performed an extensive simulation study. The following conclusions can be made:

- The OLSR HELLO messaging protocol with its default parameters can cause enormous packet loss even in small low mobility scenarios.
- From the point of view of the OLSR neighbor sensing protocol there is no connection between the speed of the node and the amount of packets lost ff the node is mobile The amount of packets lost depends on both the data rate and the frequency of the HELLO messages.

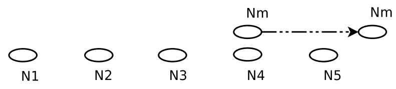

Fig. 12. N_m has connection with N_3, N_4 and N_5

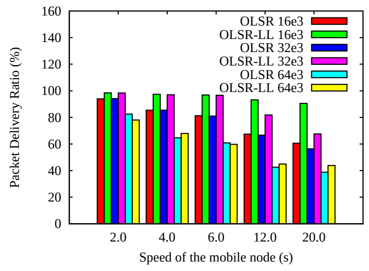

Fig. 13. Packet Delivery Ratio in a random waypoint scenario

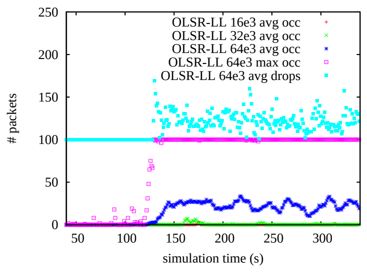

Fig. 14. Buffer occupancy measurements

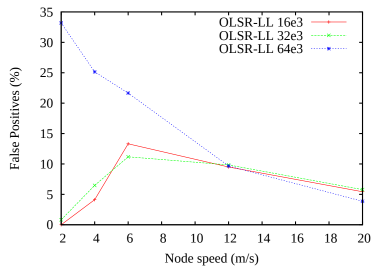

Fig. 15. % of false positive in link-layer feedback

- Link layer feedback from the 802.11 MAC layer can improve the performance, but introduces asymmetric neighbor sensing since the receiving node still needs to rely on the timeout. Feedback from the 802.11 MAC layer also becomes inaccurate when the network load increases. This has an effect on the operation of the OLSR protocol since it is the receiving node that is responsible for choosing MPRs that should advertise that they can reach the receiving node. The tunable parameter MPR coverage can be used to solve this problem.
- The Link-Layer feedback scheme works well in both the strip scenario and the generalized random waypoint scenario, without increasing the signalling overhead like Fast-OLSR.
- When the network load is high the accuracy of the L2 feedback remains acceptable, however the simple scheme is not resilient to multiple packets with the same next hop present in the link-layer queue when a link between the two neighbors dissapears.

Our results show that neighbor sensing can have a large impact on the performance of a routing protocol, especially in terms of packet loss and overhead. We feel confident that the work described in this article will prove a good guideline for future analysis of the OLSR protocol. In the future we plan to extend our work and be able to classify packet losses at a more detailed level and thus be able to tackle their origins in different steps. We are also interested in improving the simple link-layer feedback scheme, since its benefits in terms of performance for the OLSR protocol were interesting, to say the least.

Acknowledgment

This work was supported by the Fund for Scientific Research Flanders under Scientific Research Community Broadband communication and multimedia services for mobile users, by DWTC Belgium under project IAP P5/11 MOTION (Mobile Multimedia Communication Systems and Networks), by IWT under project 020152–End-to-end QoS in IP based Mobile Networks, by the European NoE EuroNGI and by the European project WIDENS.

References

[1] The Network Simulator ns-2.
[2] M. Neufeld, A. Jain, D. Grunwald, ”Nsclick: bridging network simulation and deployment”, In *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems, Atlanta, Georgia, USA*, 2002
[3] E. Kohler, R. Morris, B. Chen, J. Jannotti, M. Frans Kaashoek, ”The Click modular router”, In *ACM Transactions on Computer Systems 18(3), pages 263-297*, August 2000
[4] E. Kohler, ”The Click modular router”, Master thesis, MIT, November 2000
[5] A. Tønnesen, ”Implementing and extending the Optimized Link State Routing protocol”, Ph.D. thesis, UniK, November 2000
[6] P. Jacquet, T. Clausen., ”RFC 3626: Optimized Link State Routing Protocol (OLSR)”, Oct 2003.
[7] C. Perkins, E. Belding-Royer, S. Das, ”RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing”, July 2003.
[8] B. David, David A. Johnson, Hu Yih-Chun, ”The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)”, July 2004.
[9] M. Benzaid, P. Minet, K. Al, ”Integrating fast mobility in the OLSR routing protocol)”, *INRIA Research Rapport*, June 2002.
[10] T.H. Clausen, P. Jacquet, L. Viennot, ”Investigating the Impact of Partial Topology in Proactive MANET Routing Protocols”, In *Proceedings of The 5th International Symposium on Wireless Personal Multimedia Comminications*, 2002
[11] J. Haerri, C. Bonnet, F. Filali, ”OLSR and MPR: Mutual Dependencies and Performances”, In *Proceedings of Med-Hoc Net 2005*, June 2005
[12] A. Busson, N. Mitton, ’E Fleury, ”Analysis of the Multi-Point Relays selection in OLSR and Implications”, In *Proceedings of Med-Hoc Net 2005*, June 2005