The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'06)

TUNING OLSR

Yang Cheng Huang
University College London
Dept of Computer Science, Gower Street, London WC1E 6BT
United Kingdom
y.huang@cs.ucl.ac.uk

Saleem Bhatti
University of St. Andrews
School of Computer Science, St. Andrews, Fife, KY16 9SX
United Kingdom
saleem@dcs.st-andrews.ac.uk

Daryl Parker
Ericsson Ireland
Ericsson Software Campus, Athlone, Co Westmeath
Ireland
daryl.parker@ericsson.com

ABSTRACT

Optimised Link State Routing (OLSR) is a popular protocol for use in MANET networks. In this paper, we investigate the different impacts of tuning refresh interval timers on OLSR performance under various scenarios (varying node density and node speed). Based on the simulation results with NS2, we find that although reducing refresh intervals could improve OLSR's performance, the intervals for some message types (HELLO messages) have a bigger impact on OLSR performance than for other message types. We find that the impact of the interval timer grows with increased network mobility and node density.

I. INTRODUCTION

The Optimised Link State Routing Protocol (OLSR) [1] sends periodic HELLO messages locally to detect neighbour changes, and exchanges topology information among all the nodes of the network to discover available routes in the presence of node disconnection and node movement. Moreover, it optimises the pure link state protocol by propagating topology information via selected nodes, called *multipoint relays* (MPRs), which facilitate efficient flooding of control messages in the network.

OLSR shares some constraints with other link state (LS) protocols [3] [4]. When there is a link or node failure, the protocol takes some time to detect the failure and re-establish a consistent view of the new topology. During such a transient period, the data traffic forwarded along a path with a failed link/node will be dropped. Worse, routing loops may form because of inconsistency in routing tables and may lead to network congestion.

Research on other LS protocols (OSPF [3] and IS-IS [4]), suggests that a smaller update/refresh interval in LS routing protocols could speed up adaptation to changes at the expense of increased overhead [2]. There has been little research on how to configure the timer intervals in OLSR, or a direct assessment of how the refresh intervals and node speed variation impacts on performance. In addition, as a routing protocol for MANET, OLSR needs to consider the limited resource availability and heterogeneity of the network; increased control overhead in OLSR would be detrimental to its overall performance in data forwarding. Therefore, it is necessary to investigate the effects of tuning refresh intervals on performance under different scenarios in order to investigate how to improve performance and assess the associated overhead.

In this paper, we investigate the effects of tuning refresh intervals on OLSR routing performance. We compare the difference in performance when changing the intervals for HELLO and TC messages in simulations.

In [7] an indirect approach is used for performance analysis by considering the impact of routing protocol parameter settings on the re-routing latency. An end-to-end connectivity model is developed to explore the effect of the measured re-routing latency on end-to-end path connectivity.

We take a simpler approach than in [7], but present more detail by analysing the impacts of the parameters under different scenarios (e.g. network size and node speed); a range of values are given to the parameters, which leads to interesting observations that are non-intuitive. We measure the performance metrics directly, in simulation, including end-to-end throughput, and protocol overhead.

The remainder of this paper is organized as follows. In Section II, we briefly present the background to OLSR and show how the performance of this protocol is related to its soft-state refresh intervals. In Section III, we introduce the set-up of the simulations and then, in Section IV examine the improvements on routing performance by tuning HELLO intervals and TC intervals separately in our simulations. We conclude in Section V.

II. OPTIMISED LINK STATE ROUTING PROTOCOL

OLSR inherits the use of the link state algorithm, using *shortest path first* forwarding. It exchanges topology information with other network nodes periodically, and every node maintains the topology of the whole network. The routes are computed based on the topology information that each node holds. As a proactive protocol, routing information is available immediately when needed.

OLSR uses nodes that act as *multipoint relays* (MPRs) [5] to optimise flooding. Each node selects a set of its neighbour nodes as MPRs. In OLSR, only MPR nodes are responsible for forwarding control traffic, intended for diffusion into the entire network. MPRs provide an efficient mechanism for flooding control traffic by reducing the number of transmissions required [1].

A. State Maintenance in OLSR

In order to maintain the topology information of the whole network in the presence of mobility and failure, an

OLSR daemon needs to record and keep updated the following state information in its internal tables:

Link tuples in a *link set* keep track of the link status between the node and its neighbours. There are two types of status: *SYM* link (e.g. bi-directional) and *ASYM* link (e.g. single-directional). Each link tuple contains the interface addresses of the local node and the neighbour node (e.g. the end points of a link), and the *valid time* during which the link is considered as valid and useable [1].

Neighbour set contains *neighbour tuples* to keep track of a node's neighbour status, including *willingness* and valid time etc, while *2-hop neighbour set* records a set of *2-hop tuples* that describe symmetric links between its neighbours and the symmetric 2-hop neighbourhood.

MPR set maintains a set of neighbours that are selected as MPRs, while *MPR selector set* records a set of MPR-selector tuples and describes the neighbours that have been selected by this node as MPRs.

Topology information base (TIB) maintains topology information about the network. This information is acquired from *Topology Control* (TC) messages and is used for routing table calculations.

In the OLSR protocol, two types of control messages are used for topology information: the HELLO message and the TC message.

A node sends a HELLO message to identify itself and to report a list of neighbouring mobile nodes. From a Hello message, the mobile node receives information about its immediate neighbours and 2-hop neighbours, and selects MPRs accordingly (for details see [1]).

A TC message originates at an MPR node announcing who has selected it as an MPR. Such messages are relayed by other MPRs throughout the entire network, enabling the remote nodes to discover the links between an MPR and its selectors. Based on such information, the routing table is calculated using the shortest-path algorithm [1].

It is clear that the maintenance of the internal state information held at nodes is directly related to the exchange of HELLO and TC messages and so anything that affects when these messages are generated, such as refresh timer intervals, is likely to impact on protocol performance.

B. Soft state in OLSR

The *soft state* approach to signalling is used in OLSR. The routing state times out and is removed unless periodically refreshed by the receipt of routing updates. Soft-state signalling does not require explicit state removal or orphaned state removal when the state installer crashes since non-refreshed state will finally time-out. Also, periodic refresh messages make the system robust to node failure, to loss/corruption of refresh messages and there is no requirement for guaranteed delivery of refresh messages [8].

Soft state approaches have been widely implemented in numerous protocols [11], including RSVP, IGMP, SIP as well as OLSR. OLSR relies heavily on the soft state approach to maintain the consistency of topology information, and the consistency of routing tables amongst network nodes. So, apart from normal periodic messages, the protocol does not generate extra control traffic in response to link failure and node join/leave events.

In OLSR, the *soft state timers* have two types of usage: *message generation* and *state maintenance*. Message generation timers (HELLO and TC interval timers) are used to send periodic HELLO and TC messages, while state-maintenance timers are to keep updated the state information in OLSR internal tables and remove obsolete state by time-out (e.g. state holding timer for the neighbour set, link set and TIB).

By default, OLSR the neighbour state holding time is set to be 3 times the value of the default OLSR HELLO interval; the OLSR TIB holding time is 3 times the default value of the TC interval. TIB and link tuple timers' expiry interval equals the TIB holding time interval.

When new nodes join the network, a node detects its new neighbours with a link-sensing process by sending periodic HELLO message. When nodes leave the network, or links between nodes go down, the corresponding link state in the link set and neighbour state in the neighbour set will be removed after the state holding timers expire. In addition, periodic topology control (TC) messages help recover from loss of topology information caused by state corruption or nodes restarting.

It is clear that the internal state maintenance in each node is related directly to the refresh intervals and so changing these will impact the protocol as a whole.

III. SIMULATIONS

A. Simulation Configuration

The simulation study is conducted with the OLSR implementation from [6] which runs in version 2.9 of NS2 [8] and uses the ad-hoc networking extensions provided by CMU [9], with a radio range of 250m radius and the use of MAC/802_11 as the media access control.

We use a network consisting of n nodes: $n = 20$ to simulate a low-density network, $n = 50$ to simulate a high-density network. The nodes are randomly placed in an area of 1000m by 1000m. All simulations run for 100s.

In order to gain good confidence in the measurement results, we run the simulations 10 times for each data point to obtain the mean value, with different mobility pattern file, i.e. a different starting state for the node positions.

We use the Random Trip Mobility Model, "a generic mobility model that generalizes random waypoint and random walk to realistic scenarios" [10] and performs perfect initialisation. Unlike other random mobility models, Random Trip reaches a steady-state distribution without a long transient phase and there is no need to discard initial sets of observations.

The mean node speed, v, is between 5m/s to 30m/s, e.g. to mimic high node mobility, the node speed is uniformly distributed between 0m/s and 40m/s, yielding a mean node speed of 20m/s. The average node pause time is set to 5s.

A random distributed CBR (Constant Bit Rate) traffic model is used which allows every node in the network to be a potential traffic source and destination, as opposed to a small fixed set of nodes. The CBR packet size is fixed at 512 bytes. There are at least $n/2$ data flows that cover almost every node.

B. Metrics

In each simulation, we measure each CBR flow's throughput and control traffic overhead and then calculate the mean performance of each metric as the result of the simulation.

Throughput is considered as the most straightforward metric for the MANET routing protocols. It is computed as the amount of data transferred (in bytes) divided by the simulated data transfer time (the time interval from sending the first CBR packet to receiving the last CBR packet).

In order to gauge the routing protocol overhead, we measure both the number of routing messages, including HELLO messages and TC messages, and the number of bytes in the routing packets transmitted; *normalised routing overhead* (NRO) is computed in order to show the routing efficiency of OLSR.

Normalized routing overhead (NRO) is defined as the ratio, P_C/P_D, of the number of control packets propagated by every node in the network, P_C, to the number of data packets received by the destination nodes, P_D.

IV. Results and Discussion

We present here results and observations from our work at the time of writing. Our work is ongoing and we list some future investigations below.

1) HELLO Interval's Impact on OLSR Performance

First, we examine OLSR's performance (throughput and overhead) with different HELLO intervals, under various scenarios (e.g. node speed and node density).

Figure 1 and Figure 2 we show the performance of OLSR in a low-density network (20 nodes, n = 20); the HELLO interval, h, is set to 1s, 2s and 3s, and the TC interval, t, is set to 5s.

Figure 3 and Figure 4 show the performance of OLSR in a high-density network (n = 50); h is set to 1s, 2s and 3s, and t is set to 5s.

From Figure 1 and Figure 3 we can see:

- Reducing the HELLO interval could improve OLSR's performance.
- The impact of tuning the HELLO interval increases with increased node speed. When the network is relatively stable with less mobility, tuning HELLO interval has smaller impact than with high mobility.
- The impact of tuning the HELLO interval has no obvious relationship with network density.

From Figure 2 and Figure 4 we can see:

- A performance improvement by reducing the HELLO interval is at the expense of increased overhead.
- The overhead grows with node density; that is, the overhead in a high-density network is much larger than that in a low-density network.

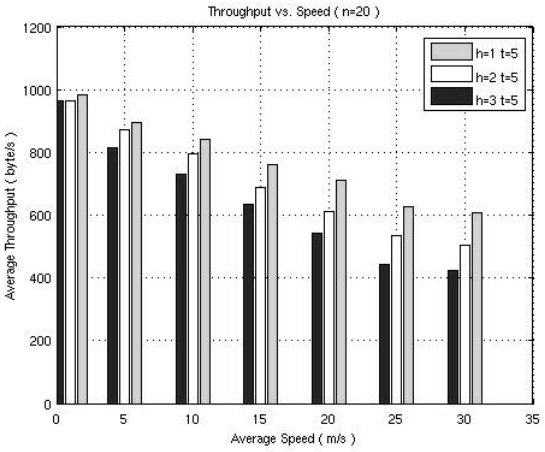

Fig. 1. Throughput vs. Speed (n=20)

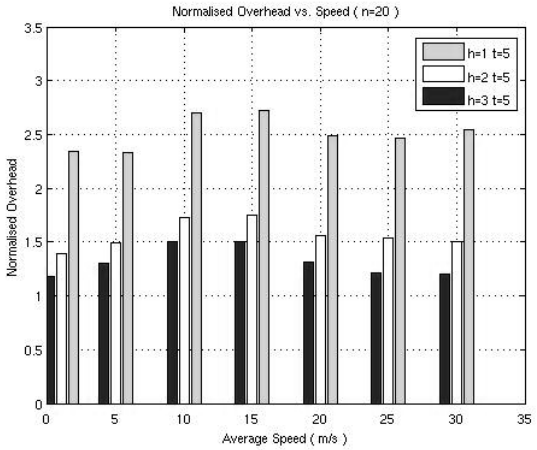

Fig. 2. NRO vs. Speed (n=20)

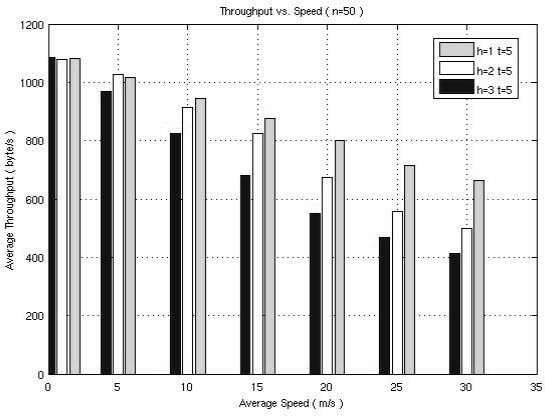

Fig. 3. Throughput vs. Speed (n=50)

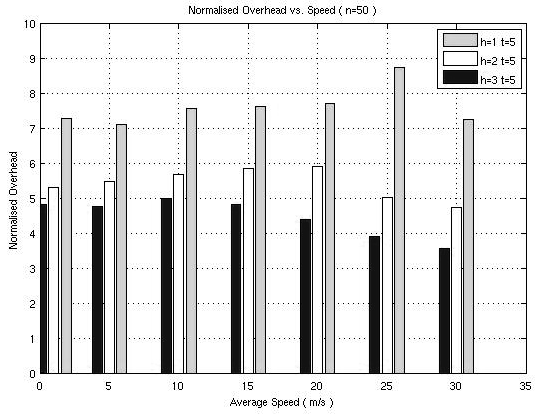

Fig. 4. NRO vs. Speed (*n*=50)

Then, we adjust the HELLO intervals in small discrete steps, to examine in more detail its impact on OLSR performance. The parameter configuration is as follows. The TC interval, *t*, is set to 5s; the HELLO, *h*, interval is increased gradually from 1s by Δh = 0.2s. The value of state holding timer intervals is adjusted correspondingly (See section II.B). Separate simulations are run for mean node speeds, v = 5m/s and v = 20m/s.

From Figure 5 & 6 and Figure 7 & 8 we can see:

- The average throughput decreases almost linearly with the increase of the HELLO interval.
- The throughput drops slightly faster in a network with high mobility (v = 20) and large density (n = 50) than with low mobility (v = 5) and small density (n = 20).
- The normalised overhead of control packets in a large-density network is much larger than in a small-density network. In addition, the overhead drops faster with the increase of the HELLO interval, in high-mobility and large-density networks than in low-mobility and small-density networks.

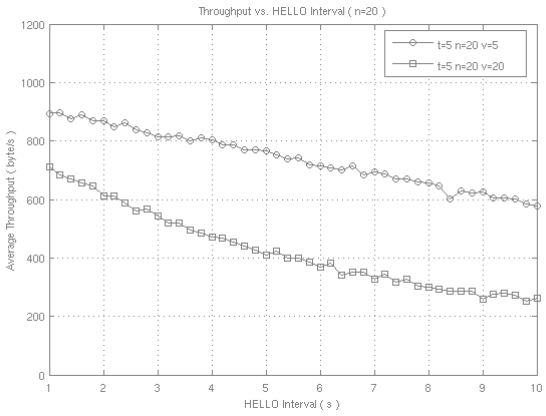

Fig. 5. Throughput vs. HELLO Interval (*n*=20)

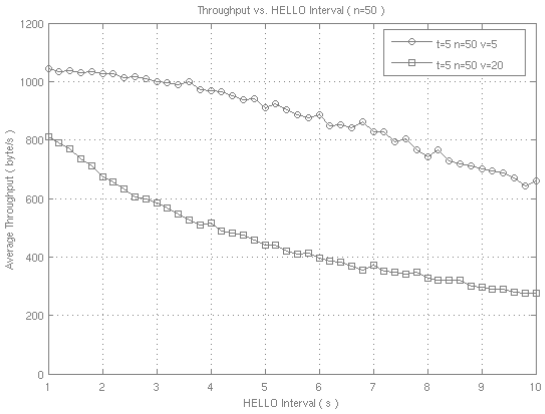

Fig. 6. Throughput vs. HELLO Interval (*n*=50)

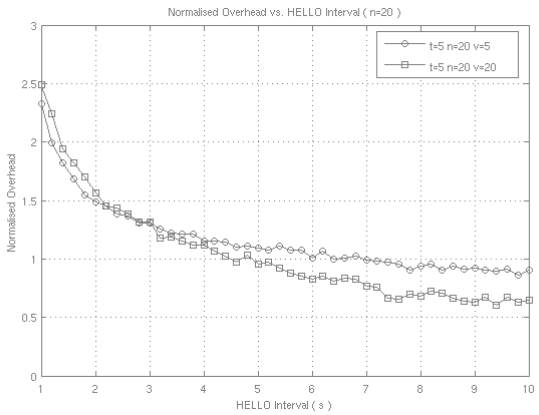

Fig. 7. NRO vs. HELLO Interval (*n*=20)

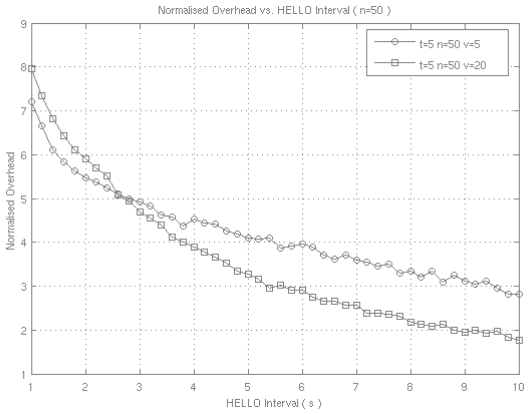

Fig. 8. NRO vs. HELLO Interval (*n*=50)

Even though the control packet overhead increases with reduced HELLO intervals, the overall throughput also increases, so one might consider continuing to reduce the HELLO interval to gain throughput, especially when mobility is high.

However, we note, from Figures 7 and 8, that there is a large increase in the number of control packets that re generated as the HELLO interval decreases. This could

lead to congestion effects in the network. Additionally, the increase in packet transmissions imposes an added computation cost and power consumption and so could lead to a quicker depletion of the battery power on a mobile device. These possibilities need investigation in order to assess in detail the possible impacts.

2) TC Interval's Impact on OLSR Performance

In order to examine the effects of tuning the TC interval on the performance OLSR, the HELLO, *h*, interval is set to 1s, 2s and 5; the TC interval is increased from 2s by by Δt = 0.2s. The value of the state holding timer intervals is adjusted correspondingly (see Section II.B).

From Figure 9 and Figure 10 we can see there is no obvious improvement on OLSR performance by tuning the TC refresh interval; it is not cost-effective to improve OLSR's performance by tuning TC intervals. Specifically, the throughput may have a very slight increase when the TC interval is reduced from 10s to 1s, while the control packet overhead increases sharply to 4-6 times.

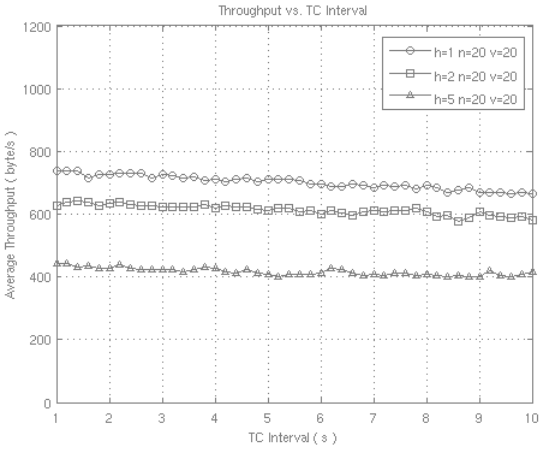

Fig. 9. Throughput vs. TC Interval

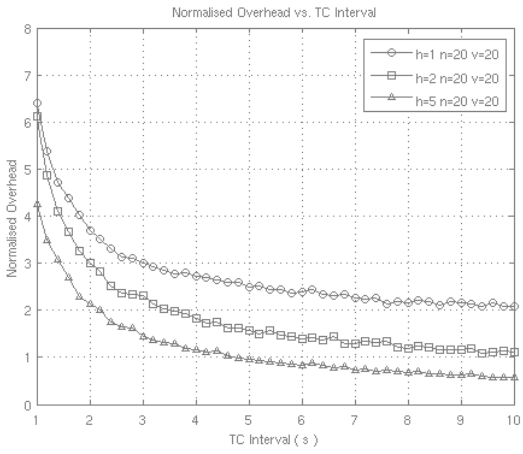

Fig. 10. NRO vs. TC Interval

V. CONCLUSIONS AND FUTURE WORK

We study the performance of the Optimised Link State Routing protocol by tuning soft state refresh intervals. Through simulations we can see OLSR routing performance is more sensitive to the value of HELLO intervals than the value of TC intervals. Although a smaller HELLO interval could speed up neighbour and link failure detection, the improvement is not linear with the decrease of the interval.

So it may be possible to tune the operation of OLSR dynamically, during operation, by measuring metrics presented in this paper, but the mechanism for performing such a dynamic tuning requires further investigation. It may also be possible to apply such analysis and tuning to other soft-state systems, in order to improve overall system performance.

The effects of the increased packet overhead (e.g. in terms of network congestion and power consumption) also need to be assessed

From Section IV we conclude that, OLSR routing performance largely depends on the value of the HELLO interval timer. That is, the protocol throughput is improved by setting up useable routes quickly and this is related to how quickly neighbouring nodes are detected. So, we are currently working on a fast neighbour detection mechanism, which could improve the performance.

VI. ACKNOWLEDGEMENTS

The authors would like to thank all those involved in the Madeira project who supported this work. Madeira is a project within the Celtic Initiative, a Eureka Cluster: http://www.celtic-madeira.org/

VII. REFERENCES

[1] T. Clausen, P. Jacquet, et. al., "Optimized link state routing protocol," RFC 3626, October 2003.

[2] M. Goyal, K. K. Ramakrishnan, W. Feng, "Achieving Faster Failure Detection in OSPF Networks", Proceedings of the IEEE International Conference on Communications (ICC 2003), Alaska, May 2003.

[3] J. Moy, "OSPF version 2", IETF Request for Comments 2328, April 1998.

[4] D. Oran, "OSI IS-IS intra-domain routing protocol", IETF Request for Comments 1142, February 1990.

[5] P. Jacquet, A. Laouiti, P. Minet, L. Viennot, "Performance of multipoint relaying in ad hoc mobile routing protocols", Networking 2002, Pise (Italy) 2002.

[6] UM-OLSR, http://masimum.dif.um.es/um-olsr/html/

[7] C. Gomez, D. Garcia, J. Paradells. "Improving Performance of a Real Ad Hoc Network by Tuning OLSR Parameters," 10th IEEE Symposium on Computers and Communications (ISCC'05), 2005.

[8] NS2, http://www.isi.edu/nsnam/ns/

[9] The Rice Monarch Project: Wireless and Mobility Extensions to ns-2, http://www.monarch.cs.rice.edu/cmu-ns.html.

[10] J. -Y. Le Boudec and M. Vojnovic, "Perfect Simulation and Stationarity of a Class of Mobility Models", IEEE Infocom 2005, Miami, FL, 2005.

[11] Ji Ping, Zihui Ge, Jim Kurose, and Don Towsley. A comparison of hard-state and soft-state protocols. In Proceedings of the ACM SIGCOMM Conference, p 251 262, Karlsruhe, Germany, Aug 2003.