State Information Accuracy in Mobile Ad-Hoc Networks Using OLSR

By

Rana Alhalimi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

Ottawa-Carleton Institute of Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario, Canada

May, 2007

The undersigned recommended to the Faculty of Graduate Studies and Research acceptance of the thesis

State Information Accuracy in Mobile Ad-Hoc Networks Using OLSR

Submitted by

Rana Alhalimi, M.C.S

in partial fulfillment of the requirements for the degree of
Master of Computer Science

Thomas Kunz
Thesis Supervisor

Frank Dehne
Director, School of Computer Science

Carleton University

May 2007

ABSTRACT

To support QoS routing, accurate state information (such as energy level and queue length) should be available and manageable. But due to bandwidth constraints, communication costs, high loss rate and the dynamic topology of MANETs, getting and keeping up-to-date state information is a very complex task, if at all feasible. In this work, we use Optimized Link State Routing (OLSR) as the underlying routing protocol.

This research reports the quantification of state information accuracy under different traffic rates. State information accuracy is defined as how far off is the believed QoS-related state information from its actual value. The results show that state information is inaccurate, especially under high traffic rates. Tuning the OLSR protocol parameters has no noticeable impact on inaccuracy levels.

Based on our inaccuracy level analysis, we proposed three additional techniques for the energy level metric and two techniques for the queue length metric as an attempt to reduce inaccuracies. We compare the different techniques against each other and against the basic OLSR. For energy level, two of our proposed techniques have shown significant improvements in inaccuracy levels. On the other hand, no improvement was observed for queue length related techniques.

TABLE OF CONTENTS

TABLE OF CONTENTS ..... **IV**

LIST OF FIGURES ..... **VII**

LIST OF TABLES ..... **IX**

ACKNOWLEDGMENTS ..... **IX**

ACRONYMS ..... **XII**

1. INTRODUCTION **1**

2. RELATED WORK **6**

2.1. AD HOC ROUTING PROTOCOLS 7

2.2. QOS SUPPORT AND OLSR 8

2.3. QOS ROUTING IN THE PRESENCE OF INACCURATE INFORMATION 10

2.4. QUEUE LENGTH FOR LOAD-BALANCED ROUTING 11

2.5. ENERGY LEVEL FOR ENERGY-AWARE ROUTING 12

3. OLSR: A PROACTIVE ROUTING PROTOCOL FOR MANETS **13**

3.1. INTRODUCTION 13

3.2. DESCRIPTION OF OLSR 14

3.2.1. OLSR MESSAGE TYPES 15
3.2.1.1. Hello Message *16*
3.2.1.2. Topology Control (TC) message *19*
3.2.2. OLSR PARAMETERS 21
3.2.2.1. Hello-Interval *21*
3.2.2.2. TC-Interval *21*
3.2.2.3. MPR-Coverage *22*
3.2.2.4. TC-Redundancy *23*

4. QUANTIFICATION OF STATE INFORMATION ACCURACY **24**

4.1. INTRODUCTION 24

4.2. QOS METRIC 24

4.2.1. QUEUE LENGTH 26
4.2.2. ENERGY LEVEL 27

4.3. QOS-RELATED STATE PROPAGATION 28

4.3.1. HELLO MESSAGE EXTENSION FORMAT 29
4.3.2. TC MESSAGE EXTENSION FORMAT 30

4.4. COLLECTING THE QOS-RELATED STATE INFORMATION 31

5. QUEUE LENGTH AND ENERGY LEVEL METRICS PERFORMANCE 35

5.1. INTRODUCTION 35

5.2. SIMULATION ENVIRONMENT 35

5.2.1. TRAFFIC MODEL 36

5.3 DATA COLLECTION MODULE 37

5.4. QUEUE LENGTH METRIC 38

5.4.1. QUEUE LENGTH OVERALL INACCURACY LEVEL 38
5.4.2. AVERAGE QUEUE LENGTH 43
5.4.3. EFFECT OF VARYING OLSR PARAMETERS ON OVERALL INACCURACY LEVEL 44
5.4.3.1. Confidence Interval Test 45
5.4.3.2. Overall Inaccuracy Level Under Different OLSR parameters 46
5.4.4. NUMBER OF HOPS INACCURACY LEVEL 47
5.4.5. KNOWLEDGE AGE INACCURACY LEVEL 52

5.5 ENERGY LEVEL METRIC 56

5.5.1. ENERGY LEVEL OVERALL INACCURACY LEVEL 57
5.5.2. EFFECT OF VARYING OLSR PARAMETERS ON OVERALL INACCURACY LEVEL 59
5.5.3. NUMBER OF HOPS INACCURACY LEVEL 60
5.5.4. OVERALL INACCURACY LEVEL UNDER HELLO1TC3 OLSR 65
5.5.5. KNOWLEDGE AGE INACCURACY LEVEL 68

5.6. CONCLUSION 69

6. QUEUE LENGTH INACCURACY IMPROVEMENT 70

6.1. INTRODUCTION 70

6.2. THRESHOLD-BASED UPDATES 70

6.3. PROBING TECHNIQUE 72

6.3.1. PROBE MESSAGE STRUCTURE 73
6.3.2. PROBE MESSAGE GENERATION AND PROCESSING 74

6.4. OVERALL INACCURACY LEVEL UNDER PROBING 77

6.4.1. THRESHOLD VALUE 77
6.4.2. PERFORMANCE MEASUREMENTS OF THE PROBING TECHNIQUE 78
6.4.3. SUMMARY OF THE PROBING TECHNIQUE WITH THE DIFFERENT THRESHOLD VALUES AND DEFAULT OLSR 83

6.5. RANDOM GUESSING VERSUS QUEUE LENGTH PROPAGATION 85

6.6. EXPONENTIAL AVERAGING 87

6.7. CONCLUSION 88

7. ENERGY LEVEL INACCURACY IMPROVEMENT ... 89

7.1. INTRODUCTION ... 89

7.2. OVERALL INACCURACY LEVEL USING GUESSING ... 89

7.3. OVERALL INACCURACY LEVEL USING PREDICTION ... 90

7.4. OVERALL INACCURACY LEVEL USING SMART PREDICTION ... 93

7.5. DEFAULT OLSR VS. GUESSING VS. PREDICTION VS. SMART PREDICTION ... 94

7.6. CONCLUSION ... 97

8. CONCLUSION AND FUTURE WORK ... 99

REFERENCES ... 103

LIST OF FIGURES

Figure 1: (a) The queue length associated with the different nodes from node a's perspective (b) The actual queue length of the nodes in the network 2

Figure 2: c is the two-hop neighbor of a 16

Figure 3: Example network with 7 nodes 17

Figure 4: Each node generates a HELLO message announcing its entire neighborhood and as a result it populates the Neighbor set and two-hop neighbor set 18

Figure 5: TC message dissemination. A message generated by B will be forwarded twice; once by C then by D 20

Figure 6: Extended Hello Message Format 30

Figure 7: Extended TC Message Format 31

Figure 8: An example to illustrate the timestamp requirements in Hello and TC messages 33

Figure 9: Overall Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters. 41

Figure 10: Confidence Interval Test. A) CI1 and CI2 overlap with the average of CI1 belongs to CI2 and the average of CI2 belongs to CI1. B) CI1 and CI2 do not overlap. C) CI1 and CI2 overlap with the average of CI1 does not belong to CI2 and/or the average of CI2 does not belong to CI1 46

Figure 11: Number of Hops Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters 50

Figure 12: Knowledge Age Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters 55

Figure 13: Knowledge Age versus Number of Hops under Different Traffic Rates 56

Figure 14: Overall Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters57

Figure 15: Number of Hops Inaccuracy Level under Different Traffic Rates and Using Default OLSR Parameters 60

Figure 16: The Impact of Varying OLSR Parameters on Number of Hops Inaccuracy Level under 0.2, 0.14, 0.09, 0.04 and 0.02 Traffic Intervals Respectively 62

Figure 17: Overall Inaccuracy Level of Default OLSR vs. Hello1TC3 OLSR 66

Figure 18: Comparison between Packet Delivery Ratio under Hello1TC3 OLSR versus Default OLSR.. 67

Figure 19: Comparison between Per Packet Delay under Hello1TC3 OLSR versus Default OLSR 67

Figure 20: Knowledge Age Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters 68

Figure 21: Probe Message Structure 74

Figure 22: Node A Wants to Send a Probe Message to Node D 75

Figure 23: Knowledge Age vs. Occurrences under Different Traffic Rates and using Default OLSR Parameters 78

Figure 24: Comparison between the Probing Technique with the Different Threshold Values and Default OLSR 83

Figure 25: Queue Length Change in 1 sec Interval under Different Traffic rates and using Default OLSR Parameters 86

Figure 26: Overall Inaccuracy Level using Default OLSR vs. Guessing vs. Prediction vs. Smart Prediction 94

Figure 27: Overall Inaccuracy Level as a Function of Time under 0.2 Traffic Interval 95

Figure 28: Overall Inaccuracy Level as a Function of Time under 0.14 Traffic Interval 96

Figure 29: Overall Inaccuracy Level as a Function of Time under 0.09 Traffic Interval 96

Figure 30: Overall Inaccuracy Level as a Function of Time under 0.04 Traffic Interval 96

Figure 31: Overall Inaccuracy Level as a Function of Time under 0.02 Traffic Interval 97

LIST OF TABLES

Table 1: Simulation Parameters 36

Table 2: Traffic Model Parameters 37

Table 3: Hello/TC Message Loss and Delay under Different Traffic Rates and using Defaults OLSR Parameters 42

Table 4: Average Knowledge age under Different Traffic Rates and using Defaults OLSR Parameters .. 42

Table 5: Overall Inaccuracy Level using Instantaneous Queue Length vs. Average Queue Length under Different Traffic Rates and using Default OLSR Parameters 44

Table 6: Overall Inaccuracy Level under Different Traffic Rates and different OLSR Parameters 47

Table 7: Number of Hops Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters 50

Table 8: Percentage of Active Nodes under Different Traffic Rates 52

Table 9: Average Nodal Energy Consumption, Hello/TC Message Loss and Delay and Average Knowledge Age under Different Traffic Rates and Using Defaults OLSR Parameters 58

Table 10: Overall Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters 58

Table 11: Overall Inaccuracy Level under Different Traffic rates and Different OLSR Parameters 59

Table 12: Number of Hops Inaccuracy Level under Different Traffic Rates and Using Default OLSR Parameters 61

Table 13: The Overhead Using TC-Interval 3 compared to MPR-Coverage 2 65

Table 14: Overall Inaccuracy Level under Default OLSR Parameters and Under a Combination of Hello-Interval 1 and TC-Interval 3 66

Table 15: Overall Inaccuracy Level under Probing with th = 7 vs. Default OLSR 79

Table 16: Percentage of Times a Node Learns More Recent Information about another Node from a Probe Message 79

Table 17: Probing Technique with th = 7 Overhead 80

Table 18: Overall Inaccuracy Level under Probing with th = 17 vs. Default OLSR and Associated Overhead with the Probing Technique 81

Table 19: Overall Inaccuracy Level under Probing with th = 21 vs. Default OLSR and Associated Overhead with the Probing Technique 82

Table 20: Overall Inaccuracy Level under Probing with th = 25 vs. Default OLSR and Associated Overhead with the Probing Technique 82

Table 21: Probe-Request to Probe-Reply Ratio 83

Table 22: Overall Inaccuracy Level under Probing with th = 21 and Reduced Probe Message Size vs. Default OLSR........ 84

Table 23: Overall Inaccuracy Levels under Exponential Averaging using threshold values of 0.9, 0.8, 0.7, 0.6, 0.5 and 0.1 under all Traffic Rates 87

Table 24: Percentage of number of times adjustment takes place under different traffic rates 93

ACKNOWLEDGMENTS

I would like to express my deep and sincere gratitude to my supervisor, Thomas Kunz, Department of systems and computer engineering. His wide knowledge and his constructive criticism have been of a great value to me. His understanding and guidance have provided a good basis for this thesis.

I would like also to express my gratitude to my co-supervisor Sivarama Dandamudi, who passed away before the completion of this work. His support and advice was a major contributor to this research.

I also owe my deepest thanks to my beloved husband and to my parents for their constant support and encouragement throughout this work.

ACRONYMS

AODV	Ad-hoc On-demand Distance Vector.
CBR	Constant Bit Rate.
CI	Confidence Interval.
DSDV	Destination Sequence Distance Vector.
DSR	Dynamic Source Routing.
ETX	Expected Number of Transmissions.
FIFO	First In First Out.
IETF	Internet Engineering Task Force.
MANET	Mobile Ad-hoc Network.
MHC	Minimum Hop Count.
MPR	Multi-Point Relay.
MTM	Medium Time.
NS2	Network Simulator 2.
NTP	Network Time Protocol.
OLSR	Optimized Link State Routing.
PPD	Per Packet Delay.
QoS	Quality of Service.
TC	Topology Control.
TCP	Transmission Control Protocol.

Chapter 1

INTRODUCTION

Optimized Link State Routing (OLSR) is a routing protocol used for Mobile Ad-Hoc Networks (MANETs) [4]. It is a best-effort proactive protocol. Proactive protocols are characterized by all nodes maintaining routes to all destinations at all times through periodic exchange of protocol messages. This gives it the advantage of having pre-computed routes available when needed. OLSR performs hop-by-hop routing where each node uses its most recent information for routing.

OLSR is highly focused on reducing the protocol overhead. And since OLSR works on periodic exchange of messages, overhead reduction can be achieved by reducing the number of messages and the size of messages as well. As a result of overhead reduction, information about QoS-related state is not propagated throughout the network.

But with the rising popularity of multimedia applications and potential commercial usage of MANETs, QoS routing in ad-hoc networks has become a very critical issue in the wireless area. To provide QoS routing, state information such as energy level, bandwidth or queue length should be available and manageable. But because the quality of wireless links changes quite frequently due to mobility and changes in surroundings, coupled with the bandwidth limitation as a result of limited available spectrum, getting and managing such knowledge is a very complex task. We define QoS routing to be any routing that is not best-effort shortest path routing. It could be maximum bottleneck bandwidth routing, minimum delay routing, minimum error rate routing, energy-efficient routing, load-balanced routing and so on.

We chose OLSR as the underlying routing protocol since it provides a systematic way to learn about the network. It also works with explicit models of the network. It can be extended and different metrics can be used [9]. It is also the chosen routing protocol in a bigger research project that this thesis is a part of.

Since OLSR works on periodic exchange of messages, QoS-related state information might not be up to date at any instance of time. Therefore, nodes might have inaccurate information about other nodes in the network, which might have a negative impact on the performance of the network.

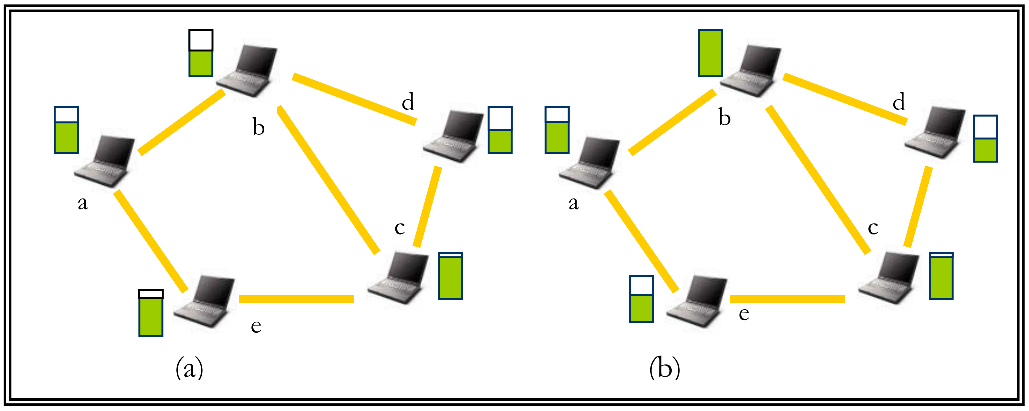


Figure 1: (a) The queue length associated with the different nodes from node a's perspective (b) The actual queue length of the nodes in the network

Figure 1 illustrates the importance of having accurate information about QoS-related state on the performance of the network. Let us assume that the bar next to the node (a node in this case is a notebook) represents the queue length associated with the node (i.e. how many packets there are in the queue). Figure 1 (a) shows what node *a* believes to be the queue length values for all the nodes in the network; whereas figure 1 (b) shows the actual queue length value associated with each node in the network. Consider the scenario by which node *a* wants to send a message to node *c*. It could send it either via *b* or *e*, however according to node's *a* view of the network, the queue of node *e* is about 100% full while node's *b* queue is only 50% full. As an attempt to achieve better performance, node *a* will send the message through *b*. Now let us look at the actual queue length information at the time node *a* sends the message. Since the time when node *a* obtained the queue length information, node *e* has released some messages from the queue and the queue became about 50% empty, on the other hand the queue of node *b* became really full. Any further messages will either be dropped due to overflow or will incur large delays. Either way the outcome is undesirable and should be avoided by having more accurate information available to all nodes in the network.

The motivation of this research is to quantify the accuracy of the QoS-related state information in ad-hoc networks under different conditions and if possible, devise techniques to reduce inaccuracies. QoS-related state information and state information are used alternatively and mean the same thing.

State information represents a QoS-related state. It could be a node attribute such as energy level and queue length, or a link attribute such as bandwidth. In this study, we are interested in node attributes, and mainly energy level and queue length. Due to energy constraints on wireless devices, energy-aware routing protocols have been widely implemented [3][20][21]. On the other hand, [6] shows that using a congestion-related metric, such as queue length, could improve the routing performance in MANETs.

State information accuracy specifies how accurate is the available QoS-related state information in terms of what their actual values are for different nodes – referred to as actual value and what other nodes believe their values are – referred to as perceived value.

The main contributions of this thesis are:

- Quantification of the inaccuracy of state information in ad-hoc networks under different traffic loads, using OLSR as the underlying routing protocol. Overall inaccuracy level is calculated as: for each pair of nodes (n1, n2) in the network such that there exists a route from n1 to n2, the average of the sum of the absolute difference between the actual state information of n2 and what n1 believes to be the state information of n2. The results show that traffic load has a significant impact on overall inaccuracy levels.

- Studying the impact of tuning the OLSR protocol parameters on the inaccuracy level of the two QoS metrics under consideration (queue length and energy level). With one exception, changing the OLSR protocol parameters had no statistically significant impact on accuracy levels.

- Suggesting and developing techniques to reduce inaccuracies. We propose two techniques for the queue length metric and three techniques for the energy level metric. We also evaluate the performance of the proposed techniques and compare them to the basic OLSR protocol performance.

  Probing and *Exponential Averaging* techniques are proposed to improve the queue length inaccuracies. Under the probing technique, whenever a node has old knowledge about another node, it sends a probe message to that node requesting its most recent information. Under the exponential averaging technique, a parameter to balance between short-term and long-term data is used. It smoothes out short term fluctuations and thus highlights long term trends. Neither technique was successful in reducing queue length inaccuracy.

  To improve the energy level inaccuracies, *Guessing*, *Prediction* and *Smart Prediction* are implemented. Under the *Guessing* mechanism, instead of propagating the energy level information, each node guesses the energy level of the other nodes based on its own energy consumption. Under the Prediction technique, a node's energy level is adjusted based on its past behavior (its own consumption rate). Smart Prediction is a modified version of the Prediction technique such that, if no behavioral pattern was known for a node then its energy levels is adjusted based on the average of all known consumption rates for other nodes. This reduces the chances of the domination of a single consumption rate. Both prediction and smart prediction can greatly reduce the energy level inaccuracy, and smart prediction outperforms prediction, in particular under high traffic loads.

The thesis is organized in the following manner: A review of the state of the art is presented in Chapter 2. Chapter 3 summarizes the core functionality of the Optimized Link State Routing Protocol (OLSR). A framework for quantifying the accuracy of the state information is presented in Chapter 4. Chapter 5 shows, using simulation, the inaccuracy level induced by OLSR under different traffic loads. It analyzes the effect of varying protocol overheads on the inaccuracy level and suggests ways to approach the inaccuracy problem for queue length and energy level. Chapter 6 presents the implementation of two techniques for improving the queue length overall inaccuracy level. It compares the performance of the proposed techniques against the basic OLSR protocol performance. For improving the energy overall inaccuracy level, Chapter 7 proposes three techniques. The performance of these techniques is analyzed and is compared to the basic OLSR protocol performance. The final chapter concludes the thesis and suggests recommendations for future work.

Chapter 2

RELATED WORK

When considering the task of QoS routing, it is important to realize that this ability is very much dependent on the accuracy of the information specifying resource availability at network nodes and links. QoS routing relies on the QoS-related state information and uses it to find paths that can satisfy certain QoS requirements.

Unfortunately, there are many reasons why state information may not accurately reflect the current state of the network. Traffic, mobility, the underlying routing protocol characteristics and environmental changes all induce constant changes in the QoS-related state. Communicating such changes in a timely fashion is expensive and, at times, not even feasible. As a result, state updates are communicated either infrequently or imprecisely.

There are two main components to the cost of timely distribution of changes in network state: the number of entities (nodes and links) generating such updates, and the frequency at which each entity generates updates. Each advertisement of a state update changes the state information on all links over which it is sent and nodes that receive it. Keeping this overhead at a minimum is desirable, if not mandatory. State updates can be propagated in a periodic fashion or in a response to a trigger (a trigger could be a significant change in the QoS-related state or an event).

As a result, the actual state of a node or a link can differ from the value known to other nodes without them being aware of it. The main consequence of this loss of accuracy in state information, in the context of QoS routing, is that nodes now need to consider not only the amount of resources (bandwidth, buffering capacity, energy level) that are available, but also the level of certainty with which these resources are indeed available. For example, a link which guarantees 10 Mb/s of available bandwidth may be more desirable than one which advertises 20 Mb/s.

2.1. Ad Hoc Routing Protocols

The IETF working group on mobile ad hoc networks has developed routing protocols optimized for different conditions. Routing protocols can be divided into reactive and proactive protocols.

In a reactive routing approach such as AODV [16] and DSR [13], a routing protocol does not take the initiative to find a route to a destination until it is required. The protocol attempts to discover routes only "on-demand" by flooding a query in the network. This type of protocols reduces control traffic overhead at the cost of increased latency in finding the route to a destination. On the other hand, proactive protocols such as OLSR [4] and DSDV [15] are based on the periodic exchange of control messages. Some messages are sent locally to enable a node to know its local neighborhood, and some messages are sent to the entire network which permits the exchange of the topology knowledge among all the nodes of the network. The proactive protocols immediately provide the required routes when needed, at the cost of bandwidth used in sending frequent periodic topology updates.

However, according to the nature of MANETs, as an infrastructure-less self-configuring network and a continuously changing topology, and the purpose of quantifying the accuracy of the QoS-related state information available to all nodes in the network about other nodes, proactive protocols are of interest. And amongst the known proactive protocols, the Optimized Link State Routing Protocol (OLSR) in particular is of interest. OLSR is a standard protocol introduced by the IETF's MANET working group. It provides a systematic way to learn about the topology which makes it a possible candidate for QoS routing [9].

The vast majority of MANETs routing protocols have used *minimum hop count (MHC)* as a metric to select paths. However, in a network with QoS guarantees, MHC might not be good criteria for path selection. It tends to favor shorter paths regardless of the path quality. In the area of QoS routing, existing shortest path based routing protocols are modified to use the QoS metric of interest.

2.2. QoS Support and OLSR

OLSR is a well-known routing protocol for ad hoc networks. It has been broadly examined [5][17], implemented and deployed [12][22]. [17] provides performance measurements over a real test-bed and concludes that even when the protocol seems to be well adapted to MANET applications it suffers from high variability of performance depending on how far apart are the nodes, and from unfairness depending on the topology and on traffic nature. It suggested that QoS features could complement the performance of the OLSR protocol.

[9] develops a QoS version of the OLSR protocol, based on link bandwidth as the QoS metric. This QoS OLSR protocol attempts to find paths with maximum bottleneck bandwidth. In order to support QoS (provide optimal bandwidth path), changes in the link-bandwidth condition must be propagated for the correct computation of the best bandwidth route. [9] evaluates the performance of this QoS OLSR model under different bandwidth change threshold values and compares it to the basic OLSR protocol performance. These threshold values define a tradeoff between the accuracy of link-bandwidth information and the additional overhead the routing protocol introduces. If the threshold is low, a relatively small change in link-bandwidth is announced, incurring a high overhead. In contrast, if the threshold value is high, only relatively large changes in link-bandwidth are advertised, reducing the overhead. Three threshold values of 20, 40 and 80% are used. The results show that amongst the proposed QoS OLSR algorithms, 20% QoS OLSR calculates the routes that are closest to the optimal routes compared to the 40 and 80% QoS OLSR. This is due to the fact that 20% QoS OLSR updates the bandwidth condition most frequently and consequently gets the most accurate bandwidth information.

[9] demonstrates that OLSR has a potential for QoS routing. It also shows that the availability of more accurate state information throughout the network, via more frequent updates, improves the performance of QoS routing. However, the results are based on the argument that 20% QoS OLSR provides the network with more accurate link state information but it

failed to investigate the level of accuracy in terms of a link actual available bandwidth and what is known to other nodes to be its available bandwidth.

[5] investigates the impact of extending topology knowledge on the OLSR protocol performance. In an OLSR network, nodes have partial topology knowledge in which only a subset of the links are known to every node due to overhead reduction. Increasing the partial topology information provides more robust and accurate topology view. It is achieved by increasing the number of links advertised and number of nodes advertising links. In OLSR this can be done by varying two protocol parameters (named MPR-coverage and TC-redundancy). In order to determine the effect of advertising redundant and more accurate topology information on the performance of the OLSR routing protocol, [5] studies the impact of increasing the MPR-coverage parameter. The two main measures of a protocol performance are packet delivery rate and per packet delay. Packet delivery rate refers to the percentage of packets that successfully reach a destination node each second, and per packet delay is the average time between a packet being sent and being received. The results in [5] show higher packet delivery rates under moderate mobility when increasing the redundancy of topological information and retransmissions provided by a higher MPR-coverage.

The research in [23] expands the work done in [5] and entirely focuses on understanding the trade-offs of increasing accurate topology knowledge. It investigates the impact of tuning the MPR-coverage and TC-redundancy parameters on the OLSR performance. It shows that delivery rates are not affected by the overhead resulted from advertising redundant information.

Both [5] and [23] focused on having more accurate information at the topology (network) level and how it affected the routing protocol performance. In other words, it studied the effect of tuning the OLSR protocol parameters on accuracy in terms of network status (existing nodes and links) and not in terms of state information available at nodes and links.

2.3. QoS Routing in the Presence of Inaccurate Information

The most relevant body of work to the problem of QoS routing in the presence of inaccurate information is a set of papers aimed at exploring state-aggregation issues and their impact on routing performance in large networks. They emphasized on developing good aggregation techniques that minimize inaccuracy in network state information, while allowing substantial reductions in the amount of state data. In particular the work in [11] addresses the information collection problem for QoS-based services in mobile environments. Specifically, it proposes a family of information collection policies that vary in the granularity at which system state information is represented and maintained. It evaluates the impact of information collection algorithms on the performance of QoS based resource provisioning. The work in [11] proposes two approaches to collecting location information for mobile applications. Fine-grained approaches maintain current location of each individual mobile client, while coarse-grained collection captures information at an aggregate level of multiple clients. [11] concludes that coarse-grained mobility information is sufficient for effective resource provisioning, whereas fine-grained mobility information introduces very high overhead. The work in [11] however does not evaluate the impact of fine-grained mobility information on resource provisioning and how it compares to coarse-grained mobility information. Since we are dealing with relatively smaller networks, our work investigates the impact of collecting fine-grained data on the accuracy of state information as an upper bound on achievable accuracy.

[1], [10] and [19] investigated the impact of inaccuracies, in the available network state and metric information, on the path selection process for flows which require QoS guarantees. In particular the work in [19] evaluates the impact of inaccurate state information on the performance and overheads of QoS routing by evaluating periodic and triggered updates. They use connection blocking as their performance measure. Connection blocking defines the percentage of times a connection request from a source to a destination fails. They draw a distinction between routing failures and setup failures. Routing failures occurs when the source cannot compute a feasible path for the new connection. In contrast, setup failures occur when the source selects a seemingly feasible path that ultimately cannot support the new connection. With a periodic update policy, larger periods substantially increase connection blocking,

ultimately outweighing the benefits of QoS routing. The results show that a purely periodic update cannot meet the dual goals of low blocking probability and low overhead in realistic networks. In contrast, experiments with triggered updates show that coarse-grained triggers do not have a significant impact on the overall blocking probability, although larger triggers shift the type of blocking from routing failures to more expensive setup failures.

However, despite the relevance to our problem, none of the prior works fully address the problem of quantifying the inaccuracy in state information. On the other hand, our work focuses on understanding the impact of how the QoS-related state is collected and how it is updated in an ad-hoc network in which no infrastructure is required.

2.4. Queue Length for Load-balanced Routing

Although many routing protocols have been proposed for MANETs, most of them have not taken load information into consideration. It is concluded in [6] that using congestion-related metric, such as queue length, could improve the routing performance in MANETs. However, the authors did not point out how to utilize this information in their paper. In [25], the authors utilized load information as the main path selection metric for routing in MANETs. They define the network load in a node as the total number of routes passing the node and its neighboring nodes. This load metric may not be accurate since the traffic along different paths may not be the same. In [14], a load-aware routing method is proposed. This protocol defines the network load of a node as the number of packets queuing in the interface of the node. Another load-aware routing protocol was proposed in [24]. It also utilizes network load information as the main path selection criterion. In their protocol, the network load in a node not only includes the local load, but also considers the load in the nodes' neighboring nodes due to the broadcast feature of the radio channels, which creates contention.

The results in [14], [25] and [24] show better performance in terms of packet delivery ratio and end-to-end delay. Our work investigates the possibility of making OLSR a load-sensitive routing protocol by evaluating how accurately individual nodes learn the current queue length for other nodes. We explore both mechanisms based solely on OLSR as well as additional techniques to further increase accuracy, as described in Chapters 5 and 6.

2.5. Energy Level for Energy-aware Routing

Due to energy constraints on wireless devices, energy-aware routing protocols have been widely implemented [3][20][21]. All the proposed protocols aim at maximizing the lifetime of the network while keeping a good protocol performance. Mostly, in all the proposed protocols, path selection is based on the remaining energy of a node such that a good candidate for the selected path should avoid nodes with low remaining energy.

In order to make better routing decisions, information about the nodes' remaining energy should be as close to the actual value as possible. Our work aims at quantifying the accuracy of the energy level information and exploring ways to improve inaccuracies. The results of this investigation are described in Chapters 5 and 7.

Chapter 3

OLSR: A PROACTIVE ROUTING PROTOCOL FOR MANETS

3.1. Introduction

An ad-hoc wireless network consists of a collection of wireless nodes, all of which may be mobile. And as opposed to conventional wireless networks which require some type of fixed infrastructure (like base stations) and some centralized administration for their operations, an ad-hoc network requires no infrastructure and is a self-configuring network.

Each node operates not only as an end-system, but also as a router to forward packets. The nodes are free to move randomly and organize themselves into a network. The nodes in an ad hoc network can dynamically join and leave the network, frequently and unpredictably. They can communicate directly to one another.

In principle, this type of network can operate as a standalone network without having any fixed infrastructure. However, this offers only a limited functionality, so usually an ad hoc network is globally connected to the larger Internet.

An ad hoc network allows direct communication between any two nodes. But due to transmission power limitations of the nodes and radio propagation conditions between these nodes, communication between two nodes often needs to be relayed through intermediate nodes. Nodes within transmission radius of one another are considered direct neighbors (referred to as 1-hop neighbors) and they can communicate directly, whereas communication between non-direct neighbors is relayed through intermediate nodes; i.e. multi-hop routing is used.

In multi-hop routing, a packet is forwarded from one node to another until it reaches the destination. Of course, appropriate routing protocols are necessary to discover routes between source and destination or even to determine the absence or presence of a path to a destination node.

3.2. Description of OLSR

The IETF's MANET working group has introduced OLSR for mobile ad hoc networks (RFC 3626) [4]. The protocol is an optimization of the classical link state routing protocol. The key concept used in the protocol is that of Multi-Point Relays (MPRs).

Each node selects a set of its neighbor nodes as MPRs. Only nodes selected as MPRs are responsible for forwarding control traffic intended for diffusion into the entire network. This technique substantially reduces the message overhead as compared to a classical flooding mechanism where every node retransmits the first copy of the messages it receives throughout the network.

In OLSR, and as a second optimization, only nodes selected as MPRs are responsible for declaring link state information in the network, hence minimizing the number of control messages flooded in the network.

As a third optimization, an MPR node may choose to declare only links between itself and its MPR selectors (nodes that have selected it as MPR). So contrary to the classical link state algorithm, instead of all links, only small subsets of links are declared. As a result, each node will have a partial knowledge of the network. Indeed, the only requirement for OLSR to provide shortest path routes to all destinations is that MPR nodes declare link state information for their MPR selectors.

Due to its proactive nature, OLSR works with a periodic exchange of messages. These messages are used for the core functionality of OLSR including neighbor detection, MPR selection, topology discovery and route calculation.

In the next section, a brief description of OLSR will be given. The basic functionality of OLSR will be addressed, supported by an example. The main points that will be looked into are:

- What are the different message types defined in the OLSR protocol
- How, using these messages, a node selects its MPRs
- How the MPRs are used to facilitate efficient flooding of control messages in the network

3.2.1. OLSR Message Types

OLSR defines a set of message types which are required for the core functionality of OLSR:

- Hello messages, performing the task of link sensing, neighbor detection and MPR signaling.
- TC messages, performing the task of topology discovery (advertisement of link states).

In order to understand how OLSR utilizes Hello and TC messages in determining the network topology, there are several terms that need to be defined. First and foremost, the terms *link sensing and neighbor detection* refer to the process by which a node attempts to find out who its neighbors are. *Neighbors* are nodes to which there exists a direct link over which data can be transmitted (i.e. nodes that can be heard by that node). A link can be characterized by the direction of the communication. Suppose a node *a* can only send data to another node *b* but *b* cannot send data to *a*, the link is said to be *asymmetric*. On the other hand, if *a* can communicate with *b* and *b* can also communicate with *a*, then the link is said to be *symmetric*. Furthermore, if the link between *a* and *b* is symmetric, *b* is said to be the *symmetric neighbor* of *a* and vice versa.

In OLSR, there is also the concept of the *two-hop neighbor*. A node *c* is considered to be a two-hop neighbor of *a* when *c* is a neighbor of *b* and *b* is also a neighbor of *a* (*a* not being *c* and *c* is not a direct neighbor of *a*).

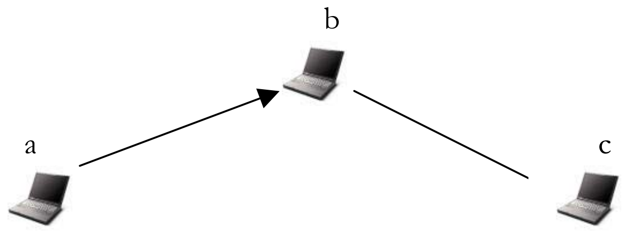


Figure 2: c is the two-hop neighbor of a

At last the term *topology discovery* refers to the process when a node attempts to find out about the other nodes in the network and how to reach them.

3.2.1.1. Hello Message

The ***neighbor sensing*** mechanism in OLSR is designed to work in this way: Every node periodically generates a Hello message advertising all 1-hop neighbors, which are at that moment known to the node, as well as the status of the link (symmetric or asymmetric) to those nodes and the type of the neighbor node (symmetric, MPR). The originator node also includes in the message its *willingness* to forward traffic on behalf of other nodes in the network. The willingness of a node is an integer value from 0 to 7 with 7 being the highest. A willingness equal to 7 indicates that a node is always willing to carry traffic on behalf of other nodes (in this case willingness = WILL_ALWAYS). On the other hand, a willingness equal to 0 indicates a node which does not wish to carry traffic for other nodes (in this case willingness = WILL_NEVER).

Whenever a node receives a Hello message, it extracts information about its neighborhood, two-hop neighborhood as well as the link status between the neighboring nodes. Each node maintains a neighborhood information base storing information about its neighbors and its two-hop neighbors (the repositories are called *Neighbor Set* and *2-Hop Neighbor Set* respectively). Each entry in the information base has a validity period, during which it is constantly being

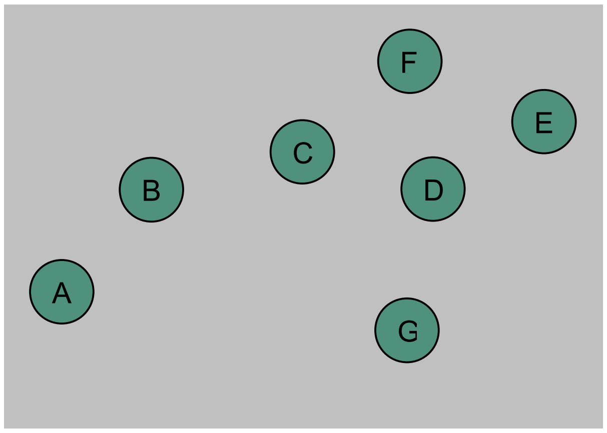
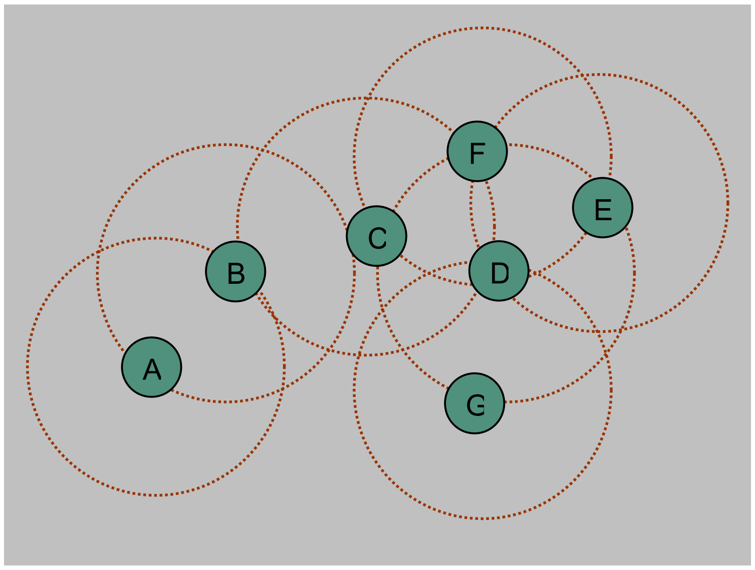


Figure 3: Example network with 7 nodes

refreshed to remain valid, or after which it expires and is automatically removed from the table.

To illustrate the neighbor detection process with an example, suppose we have a network of 7 nodes (a node could be a laptop, PDA, a cell phone….etc) as shown in Figure 3. Let us look at the topology from node *C*'s perspective: *C* will send a Hello message announcing nodes *B*, *F* and *D* as its neighbors (assuming without loss of generality that it can hear them all). Similarly, node *B* will send a Hello message announcing *A* and *C* as its neighbors. When node *C* receives *B*'s Hello message it adds (or updates, if it is already in the list) node *B* to its neighbor set and it marks it as a symmetric neighbor since *C* itself is in node's *B* Hello message (that means *B* can hear *C* and *C* received the message from *B* so it can hear it as well). It also adds *A* to its 2-hop neighbor set. Similarly, node *C* will receive a Hello message from node *F*. A Hello message from *F* will include nodes *C*, *D* and *E* as its neighbors. Node *C* will add (or update, if it is already in the list) node *F* to its neighbor set and marks it as a symmetric neighbor. At the same time node *C* will add *E* to its 2-hop neighbors, whereas *D* will not be added because it is *C*'s neighbor. Similarly node *C* will receive a Hello message from *D* resulting in adding (or updating the entries for) nodes *E* and *G* to its 2-hop neighbor set.

Figure 4 shows the transmission range of all the nodes and the Hello message generation process in which all nodes determine their neighborhood (1-hop and 2-hops neighbors).



From node C's perspective	
Neighbor set	B, F, D
Two-hop neighbor set	A, E, G

Figure 4: Each node generates a HELLO message announcing its entire neighborhood and as a result it populates the Neighbor set and two-hop neighbor set

After each node has completed its process of neighbor sensing, it still does not have enough information to compute routing information to all reachable destinations in the network, as the neighborhood information base only provides information about nodes at most two hops away.

MPR selection is the key point in OLSR. Each node independently selects its MPRs from its symmetric 1-hop neighbors such that it covers all symmetric 2-hop neighbors. The smaller the MPR set, the less overhead the protocol introduces. The proposed heuristics for MPR selection is as follows:

1. Start with an empty MPR set.

2. Add to the MPR set all 1-hop neighbors, which are the only nodes to provide reachability to a 2-hop neighbor.

3. While there exists an uncovered 2-hop neighbor:

    a. Select a 1-hop neighbor which reaches the maximum number of uncovered 2-hop neighbors. In case of a tie, select the node with higher degree.

4. As an optimization, process each node y in the MPR set in increasing order of willingness. If MPR set\{y} still covers all 2-hop neighbors and willingness of node y is different than WILL_ALWAYS, then y can be removed from the MPR set.

Following the MPR selection algorithm and continuing with the same example, we describe how node C selects its MPR set. Node C will add to its MPR set nodes B and D as they are the only 1-hop neighbors with reachability to nodes A and G. Since D covers E as well, then all 2-hop neighbors are now covered and Node C's MPR set is {B, D}.

Node	A	B	C	D	E	F	G
MPR Set	B	C	B, D	C	D	C, D	D

As described above, upon receiving a Hello message, each node locally processes the message, collects neighborhood information and runs the MPR selection algorithm to select its MPRs. Furthermore, whenever a node generates a Hello message, it includes an updated list of its entire neighborhood along with the type of link to each neighbor and the type of the neighbor as well. So if node A selected node B as an MPR, node A will announce that node B is an MPR neighbor in the Hello message. As a node receives a message showing that it has been selected by a neighbor node as an MPR, it marks the neighbor as MPR selector. Every node has an *MPR Selector Set* storing information about nodes that selected it as an MPR. The MPR selector sets for all the nodes are shown in the table below for the same network.

Node	A	B	C	D	E	F	G
MPR Selector Set	-	A, C	B, D, F	C, E, F, G	-	-	-

3.2.1.2. Topology Control (TC) message

During the ***topology discovery*** phase, MPRs are used to disseminate topology information throughout the network. This topology information will allow each node to evaluate routes to destinations in the network. Each node that has been selected as an MPR by a neighbor node will periodically generate and broadcast TC messages advertising who selected it as an MPR. Thereby a node announces to the network that it has reachability to the nodes which have selected it as MPR. MPRs also forward the TC messages they receive on behalf of the originator node.

As nodes receive TC messages from other nodes, they collect information about the topology and use this information for route calculation. Topology information is maintained at each node in a repository called *Topology Set*.

So continuing our example on the same sample network; only nodes B, C and D will be generating and forwarding TC messages as they are the only nodes selected as MPRs. Figure 5 shows the sequence of transmissions following a TC message sent by node B. A TC message from B will be forwarded twice; once by node C and again by node D. The other nodes will receive the message, process it to collect topology information and then drop it.

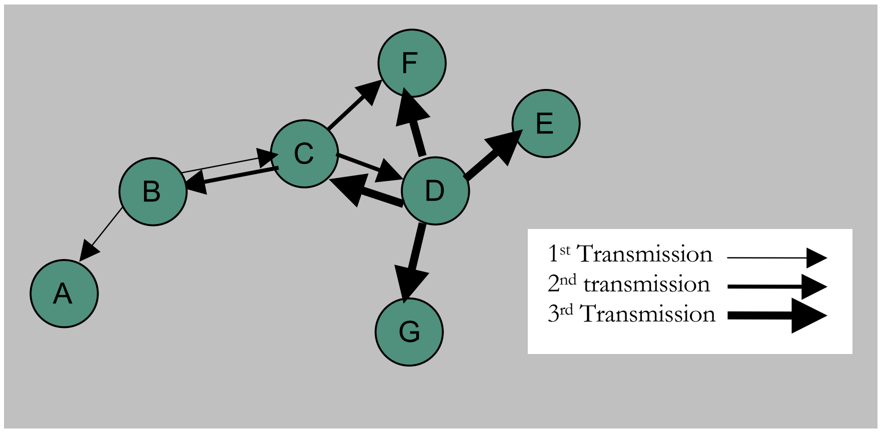


Figure 5: TC message dissemination. A message generated by B will be forwarded twice; once by C then by D

Every node maintains a routing table which allows it to route data destined for the other nodes in the network. To construct the routing table, a node uses the information in the Topology Set and runs a shortest path algorithm such as Dijkastra's algorithm. The route calculation is beyond the scope of this study so it will not be covered.

3.2.2. OLSR Parameters

OLSR is highly focused on overhead reduction. And as mentioned previously, overhead reduction can be achieved by reducing the number of messages and the amount of information advertised.

In order to control the overhead of the protocol, OLSR has defined some parameters. Following are the key parameters that control the protocol overhead.

3.2.2.1. Hello-Interval

The Hello-interval parameter represents the frequency of generating a Hello message. A Hello-interval of 2 means each node will generate a Hello message every 2 seconds. Therefore, the higher the Hello message rate, the more the overhead the protocol incurs.

Increasing the frequency of generating Hello messages leads to more frequent updates about the neighborhood and hence a more accurate view of the network. This might be required in a dynamic environment in which neighborhood information changes a lot. On the other hand, sending Hello messages less frequently leads potentially to more outdated information which can be traded off for less overhead in situations such as the network is very stable and nodes do not experience lots of changes. The default value for the Hello-interval parameter is 2.

3.2.2.2. TC-Interval

The TC-interval parameter represents the frequency of generating a TC message, i.e. a TC-interval of 5 means a TC message will be generated by MPRs every 5 seconds. The default value for the TC-interval parameter is 5.

TC messages are one of the major sources of overhead in MANETS, as they are flooded throughout the network, but they facilitate the topology discovery process. Since nodes learn about the whole topology by exchanging TC messages, the more frequently nodes generate TC messages, the better the knowledge nodes have about the topology. But at the same time, the higher the overhead the protocol incurs.

The TC-interval parameter can be set according to the characteristics of the network. If the gain in performance (more accuracy for example) is higher than the overhead incurred, then it might be a good approach to increase the frequency of generating TC messages. Whereas if the gain is low compared to the overhead, then one might consider generating TC messages at a lower rate.

3.2.2.3. MPR-Coverage

The MPR-coverage parameter represents the ability for a node to select redundant MPRs. The criterion for selecting MPRs is that all 2-hop neighbors must be reachable through at least one MPR node. Nodes should select their MPR set to be as small as possible in order to reduce protocol overhead.

Redundancy of the MPR set affects the overhead through affecting

- The amount of links being advertised, since a node will be selected by more neighbor nodes as an MPR
- The amount of nodes advertising links, since more nodes will be selected as MPRs
- The efficiency of the MPR flooding mechanism.

On the other hand, redundancy in the MPR set ensures that reachability for a node is advertised by more nodes, thus additional links are diffused to the network.

While, in general, a minimal MPR set provides the least overhead, there are situation in which overhead can be traded off for other benefits. For example, a node may decide to increase its MPR-coverage if it observes many changes in its neighbor information base caused by mobility, while otherwise keeping a low MPR-coverage.

MPR-coverage of 1 specifies that the overhead of the protocol be kept at minimum whereas MPR-coverage of *m* ensures that, if possible, a node selects its MPR set such that all 2-hop nodes are reachable through at least *m* MPR nodes. The default value for the MPR-coverage parameter is 1.

3.2.2.4. TC-Redundancy

The TC-redundancy parameter specifies, for the local node, the amount of information that may be included in the TC message.

1. TC-redundancy of 0 specifies that the advertised link set of the node is limited to the MPR Selector set.

2. TC-redundancy of 1 specifies that the advertised link set of the node is the union of the MPR Selector set and the MPR set.

3. TC-redundancy of 2 specifies that the advertised link set of the node is the full neighbor link set.

The TC-redundancy parameter affects the overhead through affecting the amount of links being advertised as well as the amount of nodes advertising links. In the case of TC-redundancy of 0, which is the default value, only MPRs generate TC messages. On the other hand, in the case of TC-redundancy of 1, all nodes generate TC messages since every node will at least advertise who its MPRs are if they are not MPR nodes. If they are an MPR node they will advertise their selectors as well. In the case of TC-redundancy of 2, all nodes will advertise their entire neighborhood.

According to the TC-redundancy parameter, the minimal set of links that any node must advertise in its TC messages is the link to its MPR selectors. But a node may decide to advertise links to its whole neighbor set. In the first case, each node will have partial topology knowledge (links to its neighbors and between its MPRs and MPR selectors). In the second case, a node will have full topology knowledge (all links). It can be shown formally though, that the partial topology knowledge is sufficient to calculate the shortest path routes from any source to any destination in the network.

Chapter 4

QUANTIFICATION OF STATE INFORMATION ACCURACY

4.1. Introduction

State information accuracy specifies how accurate the available QoS-related state information determined by the difference between the actual values and what other nodes believe their values are – referred to as perceived value – at different times.

In order to quantify the accuracy of state information, the QoS-related state needs to be propagated throughout the network. There are two ways in which QoS-related state can be propagated throughout the network. Either we define a new message type to carry the QoS-related state information, or we include it in the OLSR message types (Hello and TC messages) to be available to other nodes in the network. With the first approach, a new message type has to be defined and exchanged. This will cause more messages to be exchanged. Added to that the fact that the same gain can be obtained by including the QoS-related state information in the OLSR protocol messages, the second approach will be taken to propagate the QoS-related state information to be available to other nodes in the network.

4.2. QoS Metric

A QoS metric could be either a node-related metric (such as energy level, queue length) or a link-related metric (such as bandwidth, expected number of transmissions (ETX) [7], medium time (MTM) [2]).

In general, link-related metrics are hard to measure, usually using past behavior to predict the future. Medium time metric (MTM) [2] states that paths that minimize the total consumed medium time should be selected. In order to accomplish that, it estimates the medium time consumed by sending a packet across a link. And then it assigns a weight to that link that is directly proportional to the packet transmission time. Another metric is the expected number of transmissions (ETX) [7]. ETX states that paths with the fewest number of transmissions required to deliver a packet to a destination should be selected. The ETX of a link is the predicted number of transmissions required to send a packet over that link, including retransmissions. The ETX of a link is calculated based on measurements of packet loss.

Both the MTM and ETX metrics are based on measurements, trying to guess future packet transmission time and packet loss. But not only is the past an imperfect predictor of the future, these metrics inherently also depend on the packet size, and the metrics are typically only derived for a "typical" packet length. Consequently this induces some level of imprecision which is not desired, as the goal of this research is to quantify the inaccuracy of state information produced by the routing protocol and the network characteristics. Therefore, metrics that can be exactly (accurately) measured are preferred.

When talking about link metrics, bandwidth is the most used metric since it is one of the most critical resources in a wireless network. But supporting bandwidth as a link metric suffers from the same problems that MTM and ETX have. Shan, Chen and Nahrstedt [18] have done some work on estimating available link bandwidth, again using historical data. Ying's [9] approach of equating idle time with available bandwidth also does not work well with a random-access MAC such as IEEE 802.11, and assumes a single and constant data link rate, which is increasingly *not* true with rate-adaptive radios such as IEEE 802.11a/b/g.

Consequently, this research focuses on node-related metrics and specifically queue length and energy level. Queue length is commonly used to achieve traffic/load-balancing routing [24][25] since it is an indicator of link activity. And due to energy constraints on wireless devices, energy-aware routing protocols have been widely implemented [3][20][21]. Since these two metrics can be measured *exactly*, no imprecision problem is inherited from the nature of the metrics as was the case under link-related metrics. Under load-balancing/energy-aware routing protocols, queue length/energy level information must be available between the nodes of the network.

4.2.1. Queue Length

Packets may arrive at nodes in bursts from multiple nodes, and a node may receive more packets than it can process. Buffers (queues) hold packets until a node is able to process them. The queue length metric measures how many packets are waiting in a node's queue for processing.

There are different types of queuing methods along with queue management and packet scheduling. Packet scheduling refers to the decision process used to choose which packets should be serviced or dropped, while queue management refers to any particular discipline used to regulate the occupancy of a particular queue.

Choosing the right queuing technique and adequate queue length may be a difficult task. Configuring a queue length that is too small would result in dropping packets since the queue capacity is smaller then the arrival rate of packets. If the queue length is too large, it could introduce an unacceptable amount of latency. In general when deciding on which queuing technique to use, these questions should be answered first:

- How are incoming packets placed into queues?
- What is the order and/or method in which the networking device services its queue?
- What are the strategies for dealing with bursts of traffic and queues that overflow?

In the context of this research, there are mainly two types of queuing methods of interest. They are FIFO (first in first out) and priority queue. In FIFO queuing, packets are served in the order they are received and in case of overflow, packets are dropped from the tail of the queue. On the other hand, in priority queuing, packets are assigned a priority. Packets with higher priority are inserted at the head of the queue and they will be the first ones to be dequeued for processing. And similar to the FIFO, it implements a *tail-drop* in case of overflow.

In an OLSR network, there are two types of packets: protocol packets, which are the packets that OLSR nodes periodically exchange to determine routes; and traffic packets. However, for the operation of the network, protocol packets are of higher importance than the traffic packets. Therefore, most protocol implementations use the *Priority Queue* where control packets have higher priority than traffic packets.

As nodes receive and send packets, packets are (enqueued/dequeued) (into/from) the queue and accordingly the queue length goes up and down. Therefore queue length is a fluctuating parameter.

4.2.2. Energy Level

Generally, nodes within an ad hoc network rely on batteries for power. Energy level represents the remaining battery life for a node at any given time. Typically, every node has an initial energy value which is the level of energy the node has when it joined the network.

Communication is the main source of energy consumption for a mobile node. Nodes consume energy when transmitting data to a desired destination, when forwarding data while acting as intermediate nodes between source and destination nodes, or receiving data destined to them.

Several studies have dealt with measuring energy consumption in the wireless interfaces of mobile nodes to determine the exact sources of energy consumption in the wireless interfaces. It was found [8] that a mobile node's wireless interface not only consumes energy while communicating with other nodes, but also while in idle mode, i.e. when the node is listening to the channel but not handling packets. Following are the types of energy consumption that have been identified:

- Energy consumed while sending a packet
- Energy consumed while receiving a packet
- Energy consumed while in idle mode
- Energy consumed while in sleep mode, which occurs when the wireless interface of the mobile node is turned off (not of interest in the context of this research)

It should be noted though, that the energy consumed during sending a packet is the largest source of energy consumption of all modes. This is followed by the energy consumption during receiving a packet. Despite the fact that while in idle mode the node does not actually handle data communication operations, it was found that the wireless interface consumes a considerable amount of energy nevertheless. This amount approaches the amount that is consumed in the receive operation.

In most energy models, energy level is a monotonically decreasing parameter, as nodes continuously consume energy, even in the idle state. Although the assumption that energy level is a monotonically decreasing parameter is not realistic in practice since batteries regenerate, this assumption is correct within the constraints of the energy model we are using (i.e., the energy model implemented in NS2).

4.3. QoS-Related State Propagation

Through the exchange of OLSR control messages, each node accumulates information about the network. This information is stored according to the OLSR specifications.

However, to store the QoS-related state associated with a node, a new field has to be added to the neighborhood information base (Neighbor Set, 2-Hop Neighbor Set and MPR Selector Set) and to the topology information base (Topology Set) described in Section 3.2.1:

Neighbor Set: Each node records a set of "neighbor tuples" (N_addr, N_status, N_willingness, *N_QoS_metric*, N_time). N_addr is the address of the neighbor, N_status designates the status of the neighbor (MPR, symmetric, asymmetric), N_willingness specifies the neighbor's willingness to carry traffic on behalf of other nodes, N_QoS_metric specifies the energy level or queue length at the neighbor node, and N_time specifies the time at which this record expires and must be removed.

2-Hop Neighbor Set: Each node records a set of "2-hop tuples" (N_addr, N_2hop_addr, *N_2hop_QoS_metric*, N_time) describing links between its neighbors and the 2-hop neighborhood. N_addr is the address of a neighbor, N_2hop_addr is the address of a 2-hop neighbor, N_2hop_QoS_metric specifies the energy level or queue length of the 2-hop neighbor, and N_time specifies the time at which this record expires and must be removed.

MPR Selector Set: Each node records a set of "MPR-selector tuples" (MS_addr, *MS_QoS_metric*, MS_time). MS_addr is the address of a node that has selected the node as an MPR, MS_QoS_metric is the energy level or queue length of the MPR selector node, and MS_time specifies the time at which this record expires and must be removed.

Topology Set: For each destination in the network, each node records a set of "topology tuples" (T_dest, T_last, *T_dest_QoS_metric*, *T_last_QoS_metric*, T_seq, T_time). T_dest is the address of a node, which may be reached in one hop from the node with the address T_last. T_dest_QoS_metric and T_last_QoS_metric are the energy level or queue length of the nodes with addresses T_dest and T_last respectively. T_seq is the sequence number, and T_time specifies the time at which this record expires and must be removed.

It is worth mentioning that for describing a link-metric such as bandwidth, the same structure for the neighborhood information base would work. In this case N_QoS_metric for a neighbor tuple designates the available bandwidth on the link with that neighbor. N-2hop_QoS_metric for a 2-hop tuple will designate the bandwidth on the link between N_addr and N_2hop_addr. MS_QoS_metric for an MPR-selector tuple is the available bandwidth on the link between the node and its MPR selector with address MS_addr. For a topology tuple, there is no need for two fields describing the node metric on the destination node and the node that can reach it in one hop. Only one field that describes the QoS-related state on the link between those two nodes (the nodes with addresses T_dest and T_last) needs to be added. A topology tuple will have (T_dest, T_last, *T_QoS_metric*, T_seq, T_time).

4.3.1. Hello Message Extension Format

This section describes the Hello message extension to propagate QoS-related state.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Reserved                       | Htime         |  Willingness  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      QoS-related state                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Link Code      |   Reserved    |       Link Message Size       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Neighbor Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      QoS-related state                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Neighbor Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      QoS-related state                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                    ………………..

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Link Code      |  Reserved     |       Link Message Size       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Neighbor Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      QoS-related state                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Extended Hello Message Format

Extended Hello messages are broadcast to all one-hop neighbors, but are not relayed to further nodes. They contain not only a list of addresses of symmetric and heard neighbors, but also the most recent QoS-related state associated with those neighbors from the sender node's perspective. In addition to that, the message also contains the QoS-related state of the sender node itself at the time the message is generated. The other fields are loaded according to the OLSR specifications. The Hello message extension format is shown in Figure 6 with the new fields highlighted.

Upon receiving a Hello message, the node should update the neighbor information corresponding to the sender node as described in RFC 3626 [4] for Hello message processing. Moreover, it adds and updates QoS-related state information in the Neighbor Set, 2-hop Neighbor Set and MPR Selector Set.

4.3.2. TC Message Extension Format

Extended TC messages are broadcast and retransmitted by the MPRs in order to diffuse the messages into the entire network. It contains not only a list of addresses of a node's MPR

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|ANSN                           |              Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      QoS-related state                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Multipoint Relay Selector Address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      QoS-related state                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Multipoint Relay Selector Address                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      QoS-related state                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Extended TC Message Format

selectors, but also the QoS-related state associated with those nodes from the originator node's perspective. In addition to that, the message also contains the QoS-related state of the originator node at the time the message is generated. The TC message extension format is shown in Figure 7 with the new fields highlighted.

Upon receiving a TC message, the node should update the neighbor information corresponding to the sender node as described in RFC 3626 [4] for TC message processing. Moreover, it adds and updates QoS-related state information in the Topology Set.

4.4. Collecting the QoS-Related State Information

Having populated the neighborhood and topology information base for each node, every node can now collect what it believes to be the QoS-related state information for all other nodes in the network. At different times, every node goes through its Neighbor Set, Two-hop Neighbor Set, MPR Selector Set and Topology Set and collects the *most recent* QoS-related state information about the other nodes in the network.

The key issue here is how to collect the most recent QoS-related state information about the other nodes since the same node might have two different QoS-related state information in two different sets.

For the energy level metric, it is straight forward to determine the most recent values since energy level is a monotonically decreasing metric. Therefore, the most recent value is the smallest value. Unfortunately, this is not the case with the queue length metric, which goes up and down all the time.

A possible solution to this problem could be to add for each entry (tuple) in the information base the sequence number of the message this entry was extracted from. In this case, the tuple with the last (i.e. highest) sequence number associated with a node will have the most recent QoS-related state for this node. This solution suffers from two problems. First, sequence numbers in Hello and TC messages are not related, in other words, the numbering scheme is done separately for Hello messages and TC messages. Second, sequence numbers are done on a node basis, in other words sequence numbers are independent from different hosts.

Another approach was taken to solve this problem: a timestamp of when the data was sent out (created) is collected along with the QoS-related state. Unfortunately, timestamps are not included in the OLSR messages. Hello and TC messages have to be extended to carry timestamp information.

Moreover, it is not enough to add a timestamp field for the message only (it represents when the originator has created and sent the message out). A timestamp for all the neighbors needs to be added. It is saying that the knowledge this node has about a neighbor is extracted from a message that is received from this neighbor and the message was created with this timestamp.

Consider the following example for a network of three nodes that illustrates why this is needed:

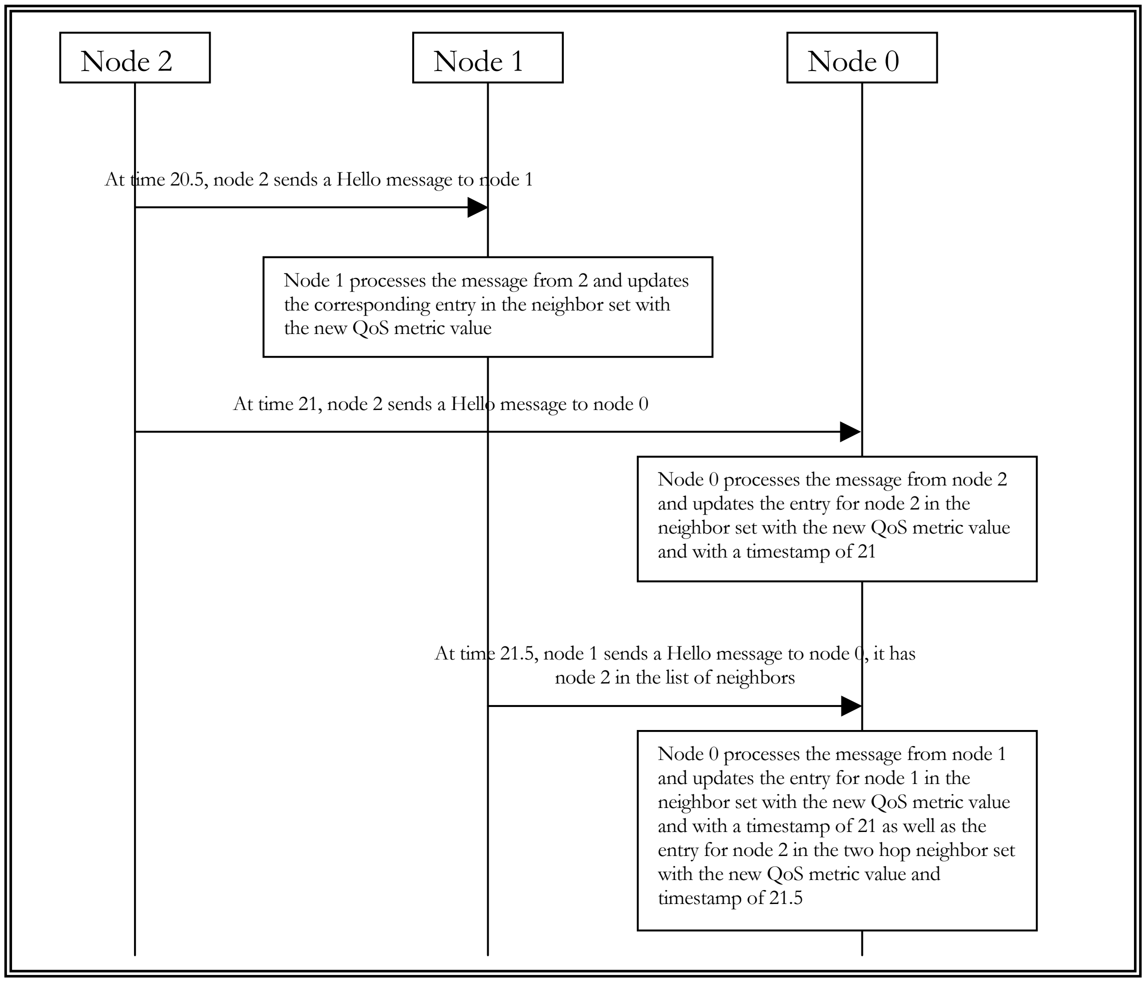

Node 2
Node 1
Node 0
At time 20.5, node 2 sends a Hello message to node 1
Node 1 processes the message from 2 and updates the corresponding entry in the neighbor set with the new QoS metric value
At time 21, node 2 sends a Hello message to node 0
Node 0 processes the message from node 2 and updates the entry for node 2 in the neighbor set with the new QoS metric value and with a timestamp of 21
At time 21.5, node 1 sends a Hello message to node 0, it has node 2 in the list of neighbors
Node 0 processes the message from node 1 and updates the entry for node 1 in the neighbor set with the new QoS metric value and with a timestamp of 21 as well as the entry for node 2 in the two hop neighbor set with the new QoS metric value and timestamp of 21.5

Figure 8: An example to illustrate the timestamp requirements in Hello and TC messages

When node 0 processes the first message, it updates the entry for node 2 in the Neighbor Set along with a timestamp of 21. And when it processes the second message, it updates the entry for node 2 in the Two-hop Neighbor Set along with a timestamp of 21.5, although the timestamp of 21.5 is true only for the originator of the message since, as seen in Figure 8, node 1 has received its knowledge about node 2 at time 20.5.

Now when node 0 collects the most recent QoS metric information, it collects the one from the Two-hop Neighbor Set although it is 1 second old rather than the one from the Neighbor Set which is 0.5 seconds old. Adding the timestamp field for each entry will solve this problem since each entry shows how old the data is.

Similar to the timestamp approach, with the sequence number approach we would need to extend the Hello and TC messages to carry for each neighbor a sequence number. It is saying that the knowledge this node has about a neighbor is extracted from the message created by this neighbor with this sequence number. Given that sequence numbers are not comparable between Hello and TC messages, this becomes a problem when collecting the most recent information about a node. Collecting the information associated with the higher sequence number will not work any more if comparing sequence numbers of different message types.

In addition, with the timestamp approach, analysis based on the age of the data can be done as well. And the fact that in both cases the Hello and TC messages have to be extended, this indeed justifies why the timestamp approach rather than the sequence number approach was taken.

One last thing, whenever we talk about networks and timing there is one concern that has to be taken into account which is, *is the network synchronized?* For the purpose of collecting the most recent QoS-related state information about a node and using the timestamp to do so, there is no need for the nodes to synchronize their clocks since we are comparing timestamps from the same node. But later in our discussion, we will use these timestamps to analyze delays and knowledge age, so it would be necessary to have clock synchronization. And since there are many clock synchronization solutions available such as Network Time Protocol (NTP), we will assume that the network is synchronized. As our analysis will deal with delays in the order of seconds, no particularly tight clock synchronization (i.e. at the level of micro seconds) is required.

Chapter 5

QUEUE LENGTH AND ENERGY LEVEL METRICS PERFORMANCE

5.1. Introduction

In this chapter, we introduce the results of evaluating the accuracy of the two QoS metrics under consideration – queue length and energy level as discussed in the previous chapter – from different perspectives. For each metric, we first assess the accuracy of state information using default OLSR parameters under different traffic rates and we then investigate the effect of tuning the protocol parameters on accuracy. We also conduct analysis to get some insights of how to reduce inaccuracies.

5.2. Simulation Environment

In order to quantify the accuracy of state information in different conditions, we will be using simulation experiments. Simulation is one of the most important methods in evaluating networking-related algorithms and protocols. The use of simulation has many benefits that make it a very powerful tool. For example, it allows repeating scenarios, experimenting with different parameters individually and investigating a variety of parameter combinations. It also allows us to understand and rectify the behavior of a system under conditions we may not be able to easily experiment with in a real environment.

In these evaluation experiments, we use the NS2 simulator. Network Simulator 2 (NS2) is a discrete-event network simulator and is the most used simulator in MANET research. We also use OOLSR. OOLSR is an OLSR implementation. It conforms to RFC 3626 [4] and can be run in NS2. It is provided by the Hipercom project [12]. We use the all-in-one NS2 package with OLSR integrated (NS2 version 2.27 with OOLSR version 0.99.15).

For all experiments, we conduct extensive NS2 simulations. For each sample point, 10 random network snapshots are generated. The simulation results presented are an average over these 10 scenarios, which reduce the chances that the results are dominated by a single scenario. The same set of 10 scenarios are used for all simulations for a given sample point, hence the different parameters are evaluated under identical conditions.

In the following, we describe the simulation conditions that we generally used in our simulation experiments. Unless otherwise specified, while discussing a certain experiment, these are the conditions under which the experiment was conducted.

Simulator Parameters	
Propagation model	TwoRayGround
Network Type	IEEE 802.11
Transmission Range	250 m
Mobility Model	Static Network
Scenario Parameters	
Topology area	1000*1000
Number of nodes	50
Simulation time (seconds)	200 seconds (with the first 50 seconds used for warming up the network)
Queue Specifications	
Queue Type	Queue/DropTail/PriQueue
Maximum Queue Limit	50
Energy Model Specifications	
Initial Energy (Joules)	1000
Transmission Power (Watt)	1.4
Receiving Power (Watt)	1.0
Idle Power (Watt)	0.83

Table 1: Simulation Parameters

5.2.1. Traffic Model

In each simulation, there are 20 communication pairs. Each source sends 128 bytes CBR packets at different intervals. We selected CBR for traffic instead of TCP to be able to perform the comparison between the algorithms under equal conditions since TCP changes the load (the number of packets it sends) based on the network conditions, which would prevent a meaningful comparison between the parameters.

The simulations are done with 5 different intervals to study the effect of low, medium and high traffic rates on the QoS metric. The intervals are 0.2, 0.14, 0.09, 0.04 and 0.02 seconds – an interval of 0.2 means a node will send a packet every 0.2 of a second (i.e. 5 packets per second).

Traffic rate is a very crucial parameter for the problem we are investigating due to the fact that the variation in both the queue length and the energy level values is tightly coupled with the traffic rate.

For the queue length, as traffic rate increases, the network becomes more congested and nodes become unable to process packets (messages) as soon as they receive them. Consequently packets need to be enqueued (put in the node's queue), waiting to be processed, and they are handled one at a time in a first in first out fashion within their priority class.

On the other hand, for the energy level, since communication is one of the main sources of energy consumption as mentioned in Section (4.2.2.), increasing the traffic rate directly impacts the variation in energy level.

Traffic Type	CBR packets
Packet Size (Bytes)	128
Number of Communicating Pairs	20
Traffic Interval	0.2, 0.14, 0.09, 0.04 and 0.02

Table 2: Traffic Model Parameters

5.3 Data Collection Module

During the simulation, a snapshot for the whole topology is taken every second. It contains information such as:

- Which other nodes a node can hear and what it believes to be their queue lengths or energy level, based on which metric we are investigating. Node n_2 is said to be heard by node n_1 if there exists a route from n_1 to n_2.
- If a node n_2 is heard by node n_1, how many hops away is it from node n_1.

- What the actual queue length or energy level for each node is.
- How many MPRs are there in the network, and which nodes have been selected as MPRs?

Also during the simulation, we keep a record of all:

- Hello and TC messages received, including when the message was received.
- Hello and TC messages sent, including when the message was sent.

In the next two sections, we will analyze in detail the queue length and energy level metrics respectively.

5.4. Queue Length Metric

As indicated in Section 4.2.1, we use the priority queue in which packets are assigned a priority. Packets with higher priority are inserted at the head of the queue and they will be the first ones to be dequeued for processing. In case of overflow, the priority queue implements a tail-drop.

In NS2, a priority queue is defined as a subclass of a drop tail queue which is a subclass of queue. The default value for the queue limit, defined as the maximum number of packets in the queue, for the Queue/DropTail/PriQueue is 50. So the queue length parameter is a value in the range between 0 and 49.

5.4.1. Queue Length Overall Inaccuracy Level

Inaccuracy level represents the average difference between a node's actual queue length and what other nodes believe its queue length is.

Overall inaccuracy level is calculated as: for each pair of nodes (n1, n2) in the network such that n1 can hear n2, the sum of the absolute difference between the actual queue length of n2 and what n1 believes to be the queue length of n2, for all time points (every second of the simulation) divided by the total number of pairs. This is done for 10 scenarios and the overall inaccuracy level is the average over the 10 scenarios.

The pseudo-code for the overall inaccuracy level calculation is given below. There are some details that need to be noticed in the algorithm:

- For the calculation of overall inaccuracy level, we ignore the first 50 seconds of the simulation to have the network stabilized and to ignore the initial conditions. We consider the information from second 50 till the end of the simulation.

- The difference between the actual queue length and the perceived queue length might be negative or positive, but since we are calculating inaccuracies, we are interested in the absolute difference between actual and perceived queue lengths.

- Finally, we are only considering pairs of nodes (n1, n2) such that n1 can hear n2. During the simulation and due to message loss and delays, some nodes get temporarily disconnected from other nodes. Therefore they are not considered part of the network as they are not used for routing. For overall inaccuracy level calculation, only visible nodes are considered, therefore, a counter is used to compute how many pairs are accounted for.

- Inaccuracy level is calculated as the sum of the absolute differences divided by number of occurrences.

```
NUMBEROFSIMULATIONS := 10;
NUMBEROFNODES := 50;
STARTINGTIME := 50;

for (index :=1 to NUMBEROFSIMULATIONS) do
begin
  inaccuracies[index] := 0;
end

for (simulationNumber := 1 to NUMBEROFSIMULATIONS) do
begin
  time := STARTINGTIME;
  difference := 0;
  occurrences := 0;
  while (! end of simulation) do
  begin
    for (each pair of connected nodes (n1, n2)) do
    begin
      difference += abs(n2 actual queue length – what n1 believes to be n2 queue length);
      occurrences ++;
    end
    time ++;
  end
  inaccuracies[simulationNumber-1] := difference/occurrences;
end

for (index := 1 to NUMBEROFSIMULATIONS) do
begin
  sumOfInaccuracies += inaccuracies[index];
end
inaccuracyLevel := sumOfInaccuracies/NUMBEROFSIMULATIONS;
```

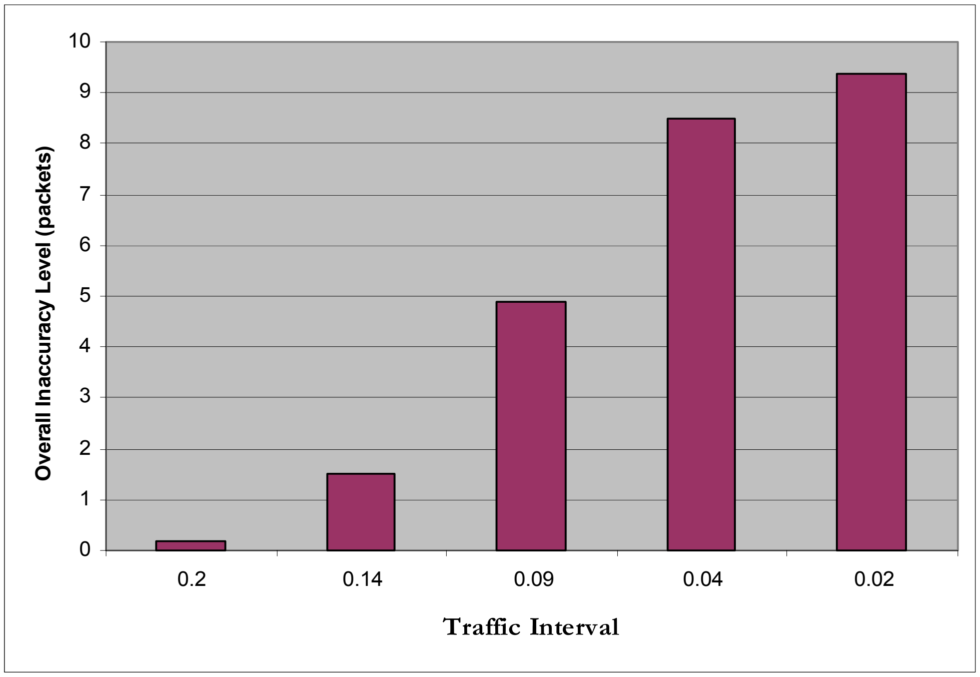


Figure 9: Overall Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters

Figure 9 shows the overall inaccuracy level under low, medium and high traffic rates using the default OLSR parameters (i.e. Hello-interval 2, TC-interval 5, MPR-coverage 1 and TC-redundancy 0).

As shown in the figure above, traffic introduces a considerable level of inaccuracy to the network. And as traffic rate increases, the level of overall inaccuracy increases. As traffic rate increases, the network becomes more congested. Since network buffers have limited capacity in terms of storage and processing of arriving packets, this affects significantly the performance of the network, causing long delays and packet loss as packets will be waiting in the queues for processing or will be dropped due to overflow. Consequently, information available to the nodes in the network becomes outdated and no longer accurate.

As a matter of fact, we have calculated Hello message delay, TC message delay, Hello message loss and TC message loss. We keep a record of all Hello and TC messages received including when the message was created and when it was received. Hello/TC message delay represents the end-to-end delay and is calculated as the sum of the difference between when the

Hello/TC message was received by a destination node and when it was created, divided by total number of Hello/TC messages received.

On the other hand, Hello/TC message loss is calculated as the percentage of Hello/TC messages received compared to the number of received messages using traffic interval 0.2 as base case. We assume that very few to no messages are lost under this base case.

Traffic Interval	Hello Message Delay (secs)	TC Message Delay (secs)	Hello Message Loss	TC Message Loss
0.2	0.016	0.114	-	-
0.14	0.12	0.542	4%	3%
0.09	0.478	1.586	9%	17%
0.04	0.866	2.127	15%	36%
0.02	1.046	2.143	18%	45%

Table 3: Hello/TC Message Loss and Delay under Different Traffic Rates and using Defaults OLSR Parameters

We have also calculated the average knowledge age. Average knowledge age is calculated as: for all pairs of nodes (n1, n2) in the network, the average of how old is the information n1 has about n2.

Traffic Interval	Average Knowledge Age (secs)
0.2	2.92
0.14	3.65
0.09	5.44
0.04	7.19
0.02	7.23

Table 4: Average Knowledge age under Different Traffic Rates and using Defaults OLSR Parameters

It is evident that increasing the traffic rate results in long delays and high message loss causing nodes to have outdated information about other nodes in the network which causes a higher level of state information inaccuracy throughout the network.

Before proceeding to studying the effect of varying the different OLSR parameters as an obvious attempt to reduce inaccuracies, there is one issue that was considered for the correctness of the queue length metric, discussed in the next subsection.

5.4.2. Average Queue Length

As mentioned in Chapter 4, whenever a node generates a Hello or TC message, it loads the message with its queue length at the time the message was generated, and we referred to it as instantaneous queue length. As nodes receive messages, they collect queue length information about other nodes in the network and we referred to it as perceived values. At each second of the simulation and for each node, we compare the node's instantaneous queue length to what other nodes believe its queue length is.

However, it might be misleading if we compare the perceived queue lengths to the instantaneous queue length since we might have a node that had a queue length of 49 most of the time. Yet at the moment we capture the queue length of that node, it dropped down to 0, although the messages generated by that node during that time will have 49 as its queue length. Therefore, it might be a better approach if instead of comparing the perceived queue lengths to instantaneous queue length, we compare the perceived queue lengths to average queue length during the previous second.

The idea here is instead of collecting *instantaneous queue length* for each node, each node collects the *average queue length* weighted by duration. So if a node has a queue length of 5 for 800ms and 6 for 200ms then average queue length is 5.2 for that period. And since we take a snapshot of the network each second of the simulation, we will have each node calculate its average queue length during the previous second. For example, if at time 21, node 0 has an average queue length of 2.5, this means that in the time period between 20 (exclusive) and 21 (inclusive), node 0 has an average queue length of 2.5.

Running simulations as described earlier in this chapter, we compared the overall inaccuracy level using instantaneous queue length to inaccuracy level introduced using average queue length. However, using average queue length rather than instantaneous queue length did not show a considerable positive impact on the inaccuracy level.

Traffic Interval	Overall Inaccuracy Level Using Instantaneous Queue Length	Overall Inaccuracy Level Using Average Queue Length
0.2	0.17	0.157
0.14	1.49	1.403
0.09	4.88	4.7
0.04	8.48	8.23
0.02	9.39	9.16

Table 5: Overall Inaccuracy Level using Instantaneous Queue Length vs. Average Queue Length under Different Traffic Rates and using Default OLSR Parameters

For a node to calculate its average queue length, it has to keep a record of queue length variation and duration of each one of them and then calculate the average each second. This process involves lots of computation which noticeably slows down the simulation time. Added to that the fact that no performance gain was achieved, for the rest of the analysis we will be using the instantaneous queue length values.

5.4.3. Effect of Varying OLSR Parameters on Overall Inaccuracy Level

As shown in Section 5.4.1, the overall inaccuracy level introduced by OLSR is quite large especially under high traffic load. But as mentioned in Chapter 3, OLSR is overly occupied on overhead reduction. Therefore OLSR defaults parameters are set in a way that achieves an acceptable performance (without intensions for QoS support) while keeping the overhead as minimum as possible.

In this section, we will investigate if it is possible to trade off cost (overhead) to gain better performance (more accurate state information). We will analyze the impact of sending more frequent Hello and TC messages (by reducing Hello and TC intervals) as well as more redundant topology information (by increasing TC-redundancy and MPR-coverage parameters). Refer to Chapter 3 for a more detailed description of these parameters.

As indicated earlier, all our results are based on simulation, such that for each experiment we run 10 scenarios (samples) to ensure that the results are not dominated by a single scenario, and each result is an average of those 10 runs.

When varying protocol parameters, we expect to see different average inaccuracy levels for the same set of scenarios. However, we need to determine whether these observed differences are "meaningful" or "significant". In such situations, a confidence interval test is performed on the set of sample data. We will first briefly describe what a confidence interval test is and then we will introduce the overall inaccuracy level under different OLSR parameters.

5.4.3.1. Confidence Interval Test

As mentioned in Section 5.3, the overall inaccuracy level is an average of 10 values corresponding to 10 independent simulation runs. By the "central limit theorem", the average of N independent, identically distributed random variables follows a normal distribution. Given that in our case, N is quite small (smaller than 30), a t-distribution with n-1 degrees of freedom is used rather than a normal distribution.

In general, the formula for calculating a confidence interval (CI) is:

$$\text{average} \pm t_{CL} * \sigma/\sqrt{n}$$

where average is the overall inaccuracy level, σ is the standard deviation, n is number of samples (equals to 10) and t_{CL} is a value that depends on the desired level of confidence. The value of t for the t-distribution depends on the confidence level and the degree of freedom (in our case 9). For our analysis, we will use a 95% confidence interval, so the t value is 2.26 according to the t-table (can be found at the end of statistics books). Actually, a 95% confidence interval is very commonly used for statistical analysis. It means that with 95% confidence, the real average lies in the confidence interval.

Once we compute the confidence intervals for the different sample points, they can be used to figure out if the result differences associated with different parameter values are statistically significant or not. For any two intervals CI1 and CI2, there are three cases:

1. CI1 and CI2 overlap with the average of CI1 belongs to CI2, and the average of CI2 belongs to CI1. In this case, the difference between the two averages is considered to be not statistically significant, or in other words the two averages are considered to be statistically the same.

2. CI1 and CI2 do not overlap. In this case, the difference between the two averages is considered to be statistically significant.

3. CI1 and CI2 overlap but the average of CI1 does not belong to CI2 and/or the average of CI2 doesn't belong to CI1. In this case the confidence interval test is not decisive and a t-test has to be done. The formula is:

$$t = (\text{average}_1 - \text{average}_2)/\sqrt{\sigma_1^2/n_1 + \sigma_2^2/n_2}$$

   Once the t is found, look it up in the standard table of significance to test whether the ratio is large enough to say that the difference is not likely to have been a chance finding. Again we are looking at 95% confidence interval ($\alpha = 0.05$) and $(n_1-1) + (n_2-1) = 18$ degrees of freedom.

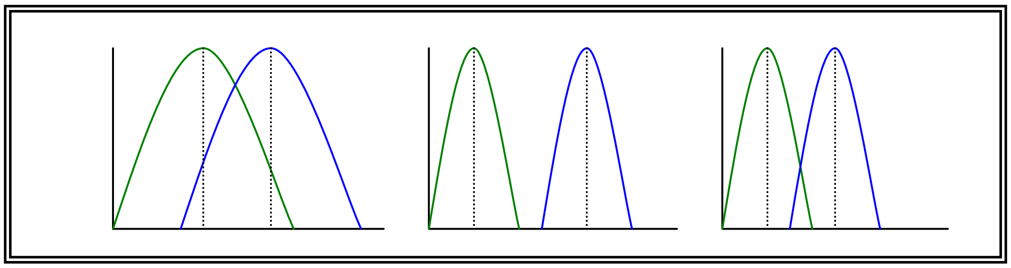


Figure 10: Confidence Interval Test. A) CI1 and CI2 overlap with the average of CI1 belongs to CI2 and the average of CI2 belongs to CI1. B) CI1 and CI2 do not overlap. C) CI1 and CI2 overlap with the average of CI1 does not belong to CI2 and/or the average of CI2 does not belong to CI1

5.4.3.2. Overall Inaccuracy Level Under Different OLSR parameters

In this section, we study the effect of each individual OLSR parameter on the overall inaccuracy level. Therefore, we vary one parameter while fixing all the other parameters to see how this parameter affects the inaccuracy level.

As shown in Table 6, a 95% confidence interval is calculated for each parameter and each traffic rate. According to the confidence interval test and the t-test, in cases where the visual CI test is not decisive, under all traffic rates, increasing the number of protocol messages including both Hello and TC messages (achieved by varying Hello-interval, TC-interval, MPR-coverage and TC-redundancy) or increasing the amount of information advertised (achieved

Traffic Interval / OLSR Parameters		0.2	0.14	0.09	0.04	0.02
Hello-interval	**2**	0.17 [0.05, 0.29]	1.49 [0.79, 2.91]	4.88 [3.66, 6.1]	8.48 [6.86, 10.09]	9.39 [7.72, 11.06]
	1	0.105 [0.06, 0.14]	1.175 [0.55, 1.79]	3.684 [2.91, 4.46]	7.16 [5.82, 8.51]	8.84 [7.22, 10.47]
TC-interval	**5**	0.17 [0.05, 0.29]	1.49 [0.79, 2.91]	4.88 [3.66, 6.1]	8.48 [6.86, 10.09]	9.39 [7.72, 11.06]
	4	0.154 [0.068, 0.24]	1.5 [0.89, 2.11]	4.81 [3.63, 6]	8.48 [7.09, 9.87]	9.77 [8.34, 11.2]
	3	0.17 [0.07, 0.27]	1.74 [1, 2.48]	5.24 [4.05, 6.42]	8.61 [7.03, 10.2]	9.82 [8.127, 11.5]
MPR-coverage	**1**	0.17 [0.05, 0.29]	1.49 [0.79, 2.91]	4.88 [3.66, 6.1]	8.48 [6.86, 10.09]	9.39 [7.72, 11.06]
	2	0.22 [0.11, 0.34]	1.94 [1.07, 2.81]	5.44 [4.19, 6.69]	8.89 [7.07, 10.71]	10.32 [8.73, 11.91]
TC-redundancy	**0**	0.17 [0.05, 0.29]	1.49 [0.79, 2.91]	4.88 [3.66, 6.1]	8.48 [6.86, 10.09]	9.39 [7.72, 11.06]
	1	0.17 [0.086, 0.25]	1.52 [0.81, 2.24]	5.15 [3.9, 6.41]	8.79 [7.17, 10.41]	9.84 [8.11, 11.57]
	2	0.17 [0.087, 0.25]	1.73 [0.89, 2.56]	4.9 [3.74, 6.06]	8.46 [6.95, 9.98]	10.15 [8.85, 11.44]

Table 6: Overall Inaccuracy Level under Different Traffic Rates and different OLSR Parameters

by varying MPR-coverage and TC-redundancy), did not have a positive impact on the overall inaccuracy level.

Clearly, the obvious approach of increasing the frequency of protocol messages or the amount of information advertised, by varying the OLSR parameters, did not seem to impact the queue length overall inaccuracy level. So to get better insights of what can be done to improve inaccuracies, we will further analyze the data based on number of hops and knowledge age.

5.4.4. Number of Hops Inaccuracy Level

Number of hops inaccuracy level represents the average difference between a node's actual queue length and what other nodes, categorized by distance in terms of number of hops to that node, believe its queue length is.

Number of hops inaccuracy level is calculated as: for all pairs of nodes (n1, n2) in the network such that n1 can hear n2, and n1 is m hops away from n2, the sum of the absolute difference between the actual queue length of n2 and what n1 believes to be the queue length of n2, for all time points (every second of the simulation) divided by number of pairs considered that are m hops apart. m is an integer in the range between 1 and 7 (inclusive). This is done for 10 scenarios and the overall inaccuracy level is the average over the 10 scenarios.

The pseudo-code for the number of hops inaccuracy level calculation is given below. There are some details that need to be noticed in the algorithm:

- All pairs of connected nodes are categorized into 7 different groups based on the distance between them. The 1st to the 6th group have all pairs that are 1-hop to 6-hops (respectively) apart.

- All nodes that are above 6-hops apart are put in the last group since the occurrences of such cases is quite small compared to nodes up to 6-hops away.

And similar to the overall Inaccuracy Level:

- The first 50 seconds of a simulation are not accounted for to get the network stabilized and to ignore initial conditions.

```
NUMBEROFSIMULATIONS := 10;
NUMBEROFNODES := 50;
STARTINGTIME := 50;

for(numberOfHops := 1 to 7)
begin
  for (simulationNumber :=1 to NUMBEROFSIMULATIONS) do
  begin
    inaccuracies[simulationNumber][numberOfHops] := 0;
  end
end

for (simulationNumber := 1 to NUMBEROFSIMULATIONS) do
begin
  time := STARTINGTIME;
  for (numberOfHops := 1 to 7 ) do
  begin
    difference[numberOfHops] := 0;
    occurrences[numberOfHops] := 0;
  end
  while (! end of simulation) do
  begin
    for (each pair of connected nodes (n1, n2)) do
    begin
      distance := distance between n1 and n2
      if distance > 6
        distance := 7;
      difference[distance] += abs(n2 actual queue length – what n1 believes to be n2 queue length);
      occurrences[distance] ++;
    end
    time ++;
  end
  for (numberOfHops := 1 to 7 ) do
  begin
    inaccuracies[simulationNumber][numberOfHops] :=
                                  difference[numberOfHops]/occurrences[numberOfHops];
  end
end

for(numberOfHops := 1 to 7)
begin
  for (simulationNumber := 1 to NUMBEROFSIMULATIONS) do
  begin
    sumOfInaccuracies[numberOfHops] += inaccuracies[simulationNumber][numberOfHops];
  end
end

for(numberOfHops := 1 to 7)
begin
  inaccuracyLevel := sumOfInaccuracies[numberOfHops]/NUMBEROFSIMULATIONS;
end
```

Following are the results on number of hops inaccuracy level under the five different traffic rates and using the default parameters since varying OLSR parameters did not seem to have an impact on overall inaccuracy level.

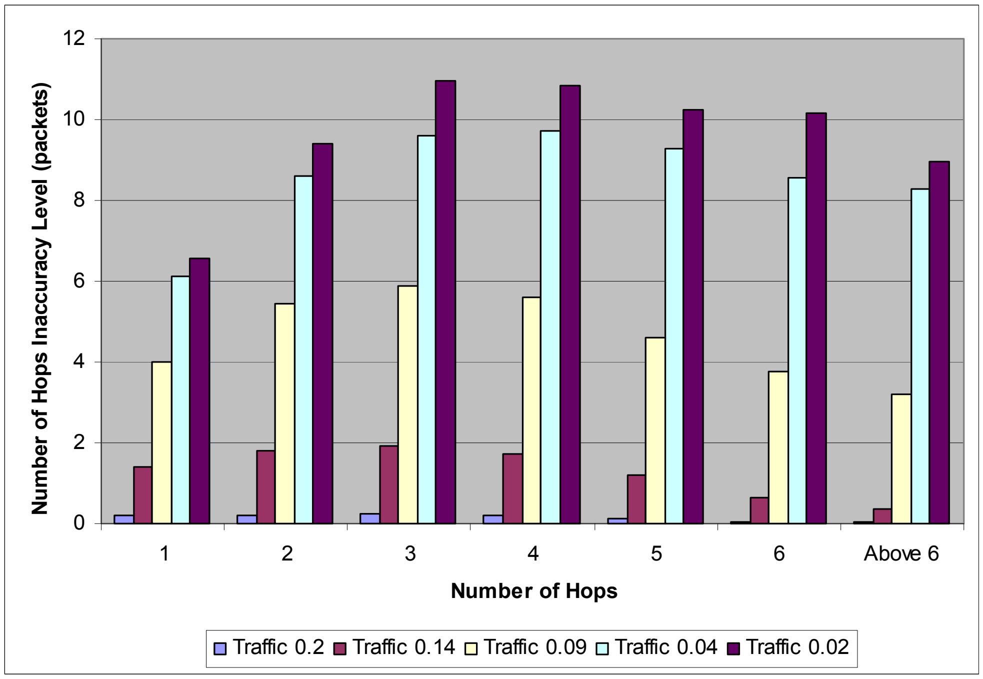


Figure 11: Number of Hops Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters

Number of Hops	Traffic 0.2	Traffic 0.14	Traffic 0.09	Traffic 0.04	Traffic 0.02
1	0.19 [0.05,0.33]	1.42 [0.77, 2.08]	4.01 [2.92,5.1]	6.14 [4.9,7.38]	6.56 [5.41,7.7]
2	0.22 [0.06, 0.38]	1.81 [0.93, 2.69]	5.45 [4.2,6.7]	8.61 [6.86,10.36]	9.4 [7.85,10.95]
3	0.23 [0.07, 0.38]	1.91 [1, 2.81]	5.88 [4.6,7.16]	9.62 [7.85,11.38]	10.97 [9.23,12.71]
4	0.19 [0.05, 0.32]	1.74 [0.94, 2.53]	5.61 [4.07,7.14]	9.72 [7.54,11.9]	10.86 [8.72,13]
5	0.13 [0.04, 0.23]	1.2 [0.67, 1.73]	4.6 [3.52,5.67]	9.27 [7.51,11.03]	10.23 [8.34,12.11]
6	0.06 [0.03, 0.09]	0.65 [0.39, 0.91]	3.75 [2.86,4.64]	8.56 [7.4,9.74]	10.15 [8.57,11.72]
Above 6	0.031 [0.01, 0.05]	0.36 [0.17, 0.56]	3.22 [2.15,4.29]	8.29 [7.19,9.4]	8.95 [7.22,10.68]

Table 7: Number of Hops Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters

The results show different trends in number of hops inaccuracy level as the traffic rate increases. Under low traffic rate, it seems that nodes that are farther apart have more accurate information than nodes which are closer. As the traffic rate increases, the difference in inaccuracy level between more distant nodes compared to nodes that are closer to each other starts to disappear and as we move towards high traffic rates, we start seeing more accurate information between closer nodes, specially nodes that are 1-hop apart, while nodes at higher hops start to have the same inaccuracy levels.

Before we explore the behavior of the different hops under different traffic rates, it is worth noticing that in a network with such a small topology area (1000*1000), to have nodes that are 6 or above hops apart on the shortest route means they have to be in the corners or along the edges of the topology. On the other hand, typically all routing happens through the nodes in the centre of the topology since they are the ones that have reachability to larger number of nodes and therefore they will be selected as MPRs.

Under low traffic rate: At first glance, it may seem strange that the higher the distance between two nodes, the more accurate the queue length information is. Intuitively, the farther the node from another node is, the less accurate the information it has. However, based on our previous observation, under low traffic rate the network is very stable and nodes on the edges of the topology are rarely selected as MPRs if at all. And even if they were selected as MPRs, not a lot of traffic goes through them and their queue length almost never changes.

As the traffic rate increases, more messages get lost or delayed and links expire. Consequently the network becomes more and more unstable and routes from a source to a destination become more congested. Nodes might need to find other routes to destinations through less congested nodes. In general, almost all the nodes participate in the routing of data traffic, and with the high volume of traffic, the change of queue lengths for those nodes is quite high.

Table 8 shows the percentage of active nodes (in terms of queue length change) based on our experiments. Active nodes are those nodes for which their queue length value changes during the simulation.

Traffic Rate	Percentage of Active Nodes
0.2	34%
0.14	50%
0.09	74%
0.04	84%
0.02	92%

Table 8: Percentage of Active Nodes under Different Traffic Rates

Also, under high traffic rate, we start seeing improvements in accuracy for closer nodes and especially 1-hop neighbors. This is due to the fact that messages do not have to travel a long distance so message loss and delays are lower. Also, 1-hop and 2-hop neighbors obtain information about a node through relatively frequent Hello messages, in addition to the less frequent TC messages.

The overall inaccuracy level analysis based on number of hops suggests that it might be possible to improve overall inaccuracy level if those far away nodes, especially under higher traffic rates, are taken care of.

5.4.5. Knowledge Age Inaccuracy Level

Knowledge age inaccuracy level represents the average difference between a node's actual queue length and what other nodes believe its queue length is, categorized by how old the knowledge is.

Node n1 learns about node n2's queue length with timestamp T_{n2}. At time T, knowledge age is determined as T- T_{n2}. We group knowledge age into 1-seconds intervals. Knowledge age inaccuracy level is calculated as: for all pairs of nodes (n1, n2) in the network such that n1 can hear n2, the sum of the absolute difference between the actual queue length of n2 and what n1 believes to be the queue length of n2, grouped by knowledge age, and divided by number of pairs considered that belong to each group. This is done for 10 scenarios and the overall inaccuracy level is the average over the 10 scenarios.

The pseudo-code for the knowledge age inaccuracy level calculation is given below. There are some details worth mentioning in the algorithm:

- The knowledge between all pairs of connected nodes is categorized into 22 different groups based on how old the data is. The 1st group has all pairs with knowledge that is 0 (inclusive) to 1 second old (exclusive). And the 2nd group has all pairs with knowledge that is 1 (inclusive) to 2 seconds old (exclusive) and so on

- All nodes that have data which is above 21 seconds old are put in the last group since in an ideal scenario it is very rare to have data that is older than 21 seconds. According to the OLSR functionality under default parameter values, it is typical to have data that is up to 6 seconds old which is the TC-interval default (every 5 seconds) plus 1 second to propagate the message. And since for each tuple we have 15 seconds holding time (3 times the TC-interval) after which the data is expired, the maximum sum is 21 seconds and then the tuple expires.

And similar to the overall Inaccuracy Level:

- The first 50 seconds of a simulation are not accounted for to get the network stabilized and to ignore initial conditions.

```
NUMBEROFSIMULATIONS := 10;
NUMBEROFNODES := 50;
STARTINGTIME := 50;

for(numOfInterval := 1 to 22) //Initialization
begin
  for (simulationNumber :=1 to NUMBEROFSIMULATIONS) do
  begin
    inaccuracies[simulationNumber][numOfIntervals] := 0;
  end
end

for (simulationNumber := 1 to NUMBEROFSIMULATIONS) do
begin
  time := STARTINGTIME;
  for (numOfIntervals := 1 to 22 ) do
  begin
    difference[numOfIntervals] := 0;
    occurrences[numOfIntervals] := 0;
  end
  while (! end of simulation) do
  begin
    for (each pair of connected nodes (n1, n2)) do
    begin
      timeDifference := how old the knowledge n1 has about n2 is
      if timeDifference >= 21
        timeDifference := 21;
      difference[floor(timeDifference)] += abs(n2 actual queue length – what n1 believes to be n2
      queue length);
      occurrences[floor(timeDifference)] ++;
    end
    time ++;
  end
  for (numOfIntervals := 1 to 22 ) do
  begin
    inaccuracies[simulationNumber][numOfIntervals] :=
                                   difference[numOfIntervals]/occurrences[numOfIntervals];
  end
end

for(numOfIntervals := 1 to 22)
begin
  for (simulationNumber := 1 to NUMBEROFSIMULATIONS) do
  begin
    sumOfInaccuracies[numOfIntervals] += inaccuracies[simulationNumber][numOfIntervals];
  end
end

for(numOfIntervals := 1 to 22)
begin
  inaccuracyLevel := sumOfInaccuracies[numOfIntervals]/NUMBEROFSIMULATIONS;
end
```

Figure 12 illustrates the results of knowledge age inaccuracy level under the five different traffic rates and using the default OLSR parameters.

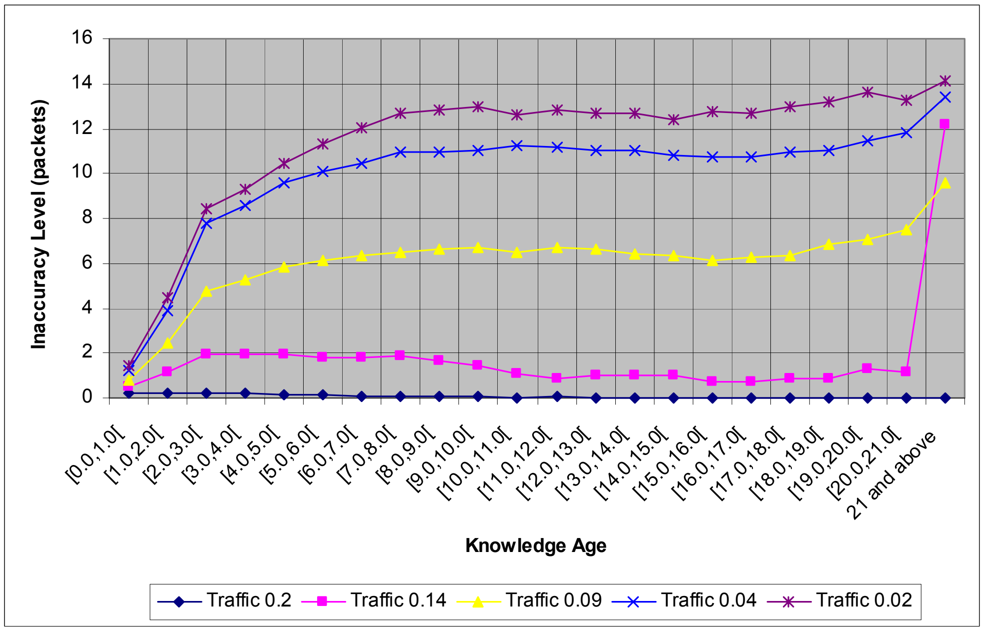


Figure 12: Knowledge Age Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters

Under higher traffic rates the results correspond to our expectations: the older the knowledge is, the less accurate it is. On the contrary, under lower traffic rates, the older the knowledge is the more accurate it is.

The relation between knowledge age and overall inaccuracy level, especially under low traffic rates, can be easily understood if we look at the relation between number of hops and knowledge age shown in Figure 13. It shows that under all traffic rates the farther a node is the older the knowledge available about it is. Under low traffic rates, the older the knowledge is the farther the node is and based on the observation we made earlier in Section 5.4.4, the farther the node is the less active it is (since it is located on the edges and not too much traffic passes through it). Therefore, the queue length changes relatively infrequently and consequently the more accurate the knowledge is.

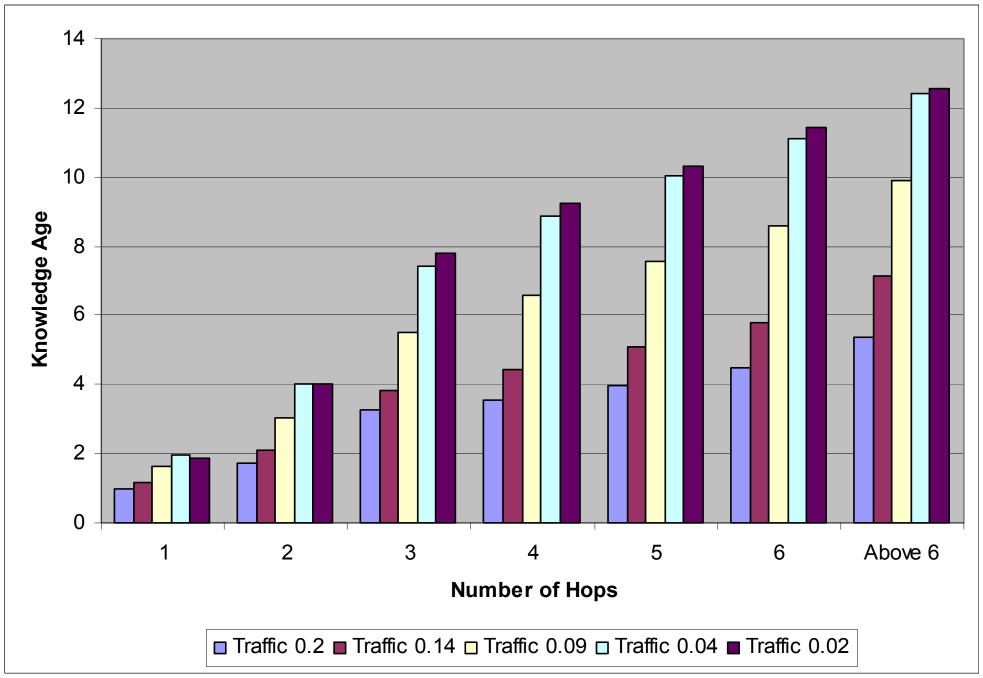


Figure 13: Knowledge Age versus Number of Hops under Different Traffic Rates

Based on the discussions in Section 5.4.4 and 5.4.5, both the number of hops and knowledge age analysis suggest that in order to improve the overall inaccuracy level, any proposed strategy has to take care of farther nodes or equivalently nodes with old knowledge. We will explore such strategies in Chapter 6.

5.5 Energy Level Metric

As discussed in Chapter 4, communication is the main source of energy consumption for an ad-hoc node. Nodes consume energy when transmitting, receiving a packet and even while listening to the medium.

It was found that the amount of energy consumed in transmitting a packet is the largest between them all, followed by the energy consumed in receiving a packet. Also, it was found that nodes consume a considerable amount of energy in the idle state although they are not handling any packets and the energy consumed is close to the energy consumed in receiving a packet.

NS2 implements an energy model which allows us to assign to a node an initial energy (initialEnergy), transmission power (txPower), receiving power (rxPower) and idle power (idlePower). Table 1 shows the energy model values used in our experiments. The values are based on the energy consumption studies presented in [8].

5.5.1. Energy Level Overall Inaccuracy Level

Inaccuracy level represents the average difference between a node's actual energy level and what other nodes believe its energy level is. It is calculated exactly the same way it was done for the queue length metric in Section 5.4.1. Note that absolute value of the difference is not needed since a node never has a lower perceived value, compared to actual value, as the energy level is a monotonically decreasing metric.

As shown in Figure 14, traffic introduces a considerable level of inaccuracy to the network and as traffic rate increases, the level of overall inaccuracy increases.

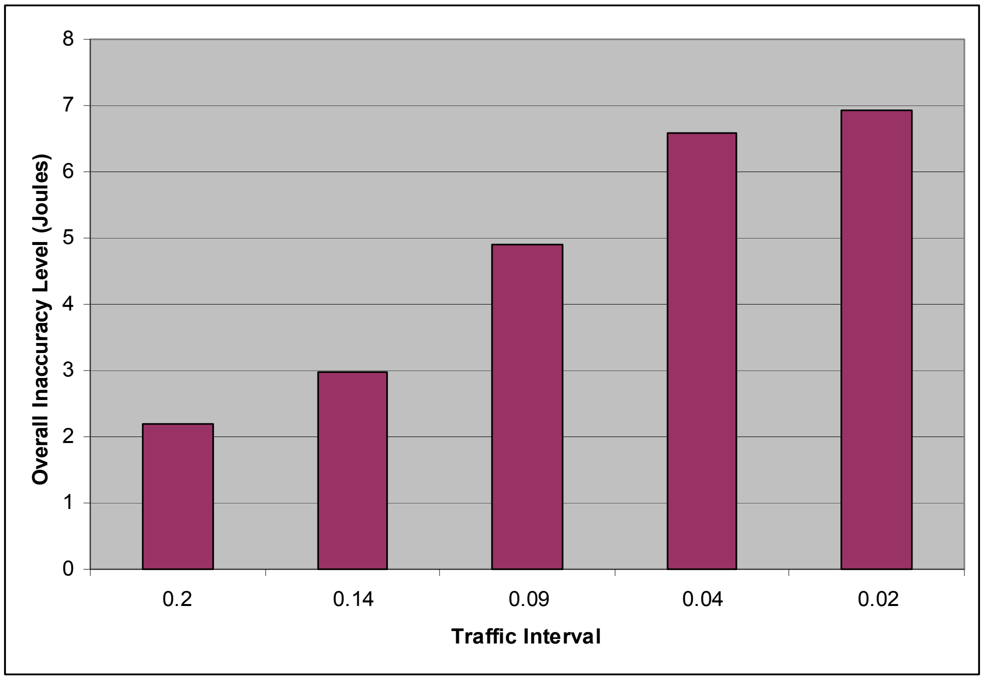


Figure 14: Overall Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters

Traffic rate contributes to overall inaccuracy level increase in different ways:

- As nodes send/receive traffic at a higher rate, they consume energy at a higher rate.
- On the other hand, more traffic causes larger delays and packets loss.

Consequently, state information available to the nodes in the network becomes outdated and no longer accurate. These observations are confirmed by our experiments. Table 9 shows the average nodal energy consumption, Hello/TC message delay and Hello/TC message loss as well as average knowledge age under the different traffic rates.

Traffic Interval	Nodal Energy Consumption (Joules/ Sec)	Hello Message Delay (secs)	TC Message Delay (secs)	Hello Message Loss	TC Message Loss	Average Knowledge Age (secs)
0.2	0.926	0.016	0.114	-	-	2.94
0.14	0.948	0.12	0.542	4%	3%	3.68
0.09	0.956	0.478	1.586	9%	17%	5.46
0.04	0.965	0.866	2.127	15%	36%	7.03
0.02	0.971	1.046	2.143	18%	45%	7.06

Table 9: Average Nodal Energy Consumption, Hello/TC Message Loss and Delay and Average Knowledge Age under Different Traffic Rates and Using Defaults OLSR Parameters

One last observation that should be mentioned here, according to a confidence interval test, the difference between the overall inaccuracy level associated with 0.04 and 0.02 traffic intervals is considered to be not statistically significant as shown in Table 10. This is due to the fact that under high traffic loads, the network is quite saturated in a sense that it almost reached its traffic capacity and excess traffic is being discarded.

Traffic Interval	Overall Inaccuracy Level	Confidence Interval
0.2	2.6225	[2.37,2.87]
0.14	3.4287	[2.96,3.9]
0.09	5.1793	[4.8,5.56]
0.04	6.7602	[5.94,7.58]
0.02	6.839	[5.86,7.81]

Table 10: Overall Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters

5.5.2. Effect of Varying OLSR Parameters on Overall Inaccuracy Level

Traffic Interval / OLSR Parameters		0.2	0.14	0.09	0.04	0.02
Hello-interval	**2**	2.6225 [2.37,2.87]	3.4287 [2.96,3.9]	5.1793 [4.8,5.56]	6.7602 [5.94,7.58]	6.839 [5.88,7.81]
	1	2.213 [1.99,2.43]	2.956 [2.47,3.44]	4.846 [4.36,5.33]	6.992 [6.16,7.82]	7.366 [6.37,8.36]
TC-interval	**5**	2.6225 [2.37,2.87]	3.4287 [2.96,3.9]	5.1793 [4.8,5.56]	6.7602 [5.94,7.58]	6.839 [5.88,7.81]
	4	2.196 [1.95,2.43]	2.978 [2.49,3.46]	5.014 [4.59,5.44]	6.774 [5.9,7.64]	6.896 [5.76,8.03]
	3	1.981 [1.76,2.2]	2.865 [2.38,3.35]	5.097 [4.66,5.53]	7.004 [6.2,7.8]	7.273 [6.15,8.39]
MPR-coverage	**1**	2.6225 [2.37,2.87]	3.4287 [2.96,3.9]	5.1793 [4.8,5.56]	6.7602 [5.94,7.58]	6.839 [5.88,7.81]
	2	2.241 [2.05,2.43]	2.888 [2.47,3.31]	4.718 [4.22,5.21]	6.925 [5.99,7.86]	6.927 [5.89,7.96]
TC-redundancy	**0**	2.6225 [2.37,2.87]	3.4287 [2.96,3.9]	5.1793 [4.8,5.56]	6.7602 [5.94,7.58]	6.839 [5.88,7.81]
	1	2.283 [2.06,2.5]	3.062 [2.6,3.52]	5.014 [4.51,5.52]	6.81 [5.98,7.64]	6.832 [5.9,7.77]
	2	2.143 [1.98,2.3]	3.005 [2.51,3.5]	4.89 [4.47,5.31]	6.79 [6.01,7.57]	7.257 [6.18,8.33]

Table 11: Overall Inaccuracy Level under Different Traffic rates and Different OLSR Parameters

As shown in Table 11, a 95% confidence interval is calculated for each parameter under the different traffic rates. Increasing the number of protocol messages including both Hello and TC messages (achieved by varying Hello-interval, TC-interval, MPR-coverage and TC-redundancy) or increasing the amount of information advertised (achieved by varying MPR-coverage and TC-redundancy), certainly improved the overall inaccuracy level under low traffic rate.

Under medium to high traffic rate, a 95% confidence interval test suggested that the difference between the overall accuracy levels under the different OLSR parameters is not statistically significant. But ignoring the CI test and taking a closer look at the inaccuracy levels values, under medium traffic rate (traffic intervals 0.14 and 0.09), it shows that there is a trend towards better inaccuracy levels when varying the OLSR parameters. On the other hand, under high traffic rate (traffic interval of 0.04 and 0.02), the trend is towards less accurate energy levels when varying the OLSR parameters.

The results observed here are a direct consequence of the increased level of congestion in the network which results in high message loss and delay and hence less accurate state information.

The obvious approach of increasing the frequency of protocol messages or the amount of information advertised improves the energy overall inaccuracy level under low traffic rates only. Therefore, further analysis based on the number of hops and knowledge age is conducted to get better insights of what can be done to improve inaccuracies under higher traffic rates or even improve it further under low traffic rates.

5.5.3. Number of Hops Inaccuracy Level

Number of hops inaccuracy level represents the average difference between a node's actual energy level and what other nodes, categorized by distance in terms of number of hops to that node, believe its energy level is. It is calculated exactly the same way it was done for the queue length metric in Section 5.4.3.

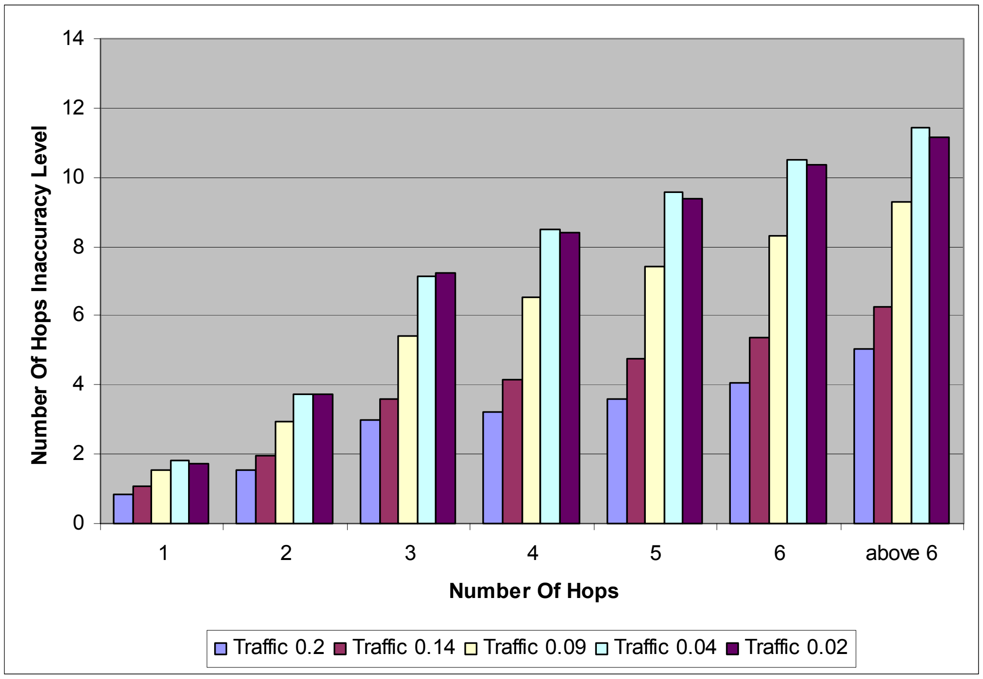


Figure 15: Number of Hops Inaccuracy Level under Different Traffic Rates and Using Default OLSR Parameters

Number of Hops	Traffic 0.2	Traffic 0.14	Traffic 0.09	Traffic 0.04	Traffic 0.02
1	0.857 [0.824,0.891]	1.054 [0.962,1.145]	1.539 [1.332,1.747]	1.813 [1.558,2.069]	1.738 [1.537,1.939]
2	1.537 [1.437,1.636]	1.937 [1.7,2.174]	2.946 [2.55,3.34]	3.736 [3.184,4.288]	3.740 [3.227,4.253]
3	2.971 [2.825,3.117]	3.612 [3.23,3.994]	5.431 [4.925,5.937]	7.118 [6.261,7.975]	7.241 [6.454,8.027]
4	3.235 [3.013,3.457]	4.173 [3.631,4.714]	6.525 [5.936,7.115]	8.475 [7.463,9.486]	8.410 [7.491,9.328]
5	3.588 [3.303,3.872]	4.745 [4.137,5.352]	7.437 [6.882,7.991]	9.581 [8.56,10.603]	9.369 [8.46,10.278]
6	4.052 [3.69,4.413]	5.382 [4.663,6.1]	8.302 [7.692,8.913]	10.483 [9.48,11.483]	10.341 [9.38,11.305]
above 6	5.047 [4.565,5.528]	6.258 [5.432,7.084]	9.294 [8.629,9.958]	11.442 [10.42,12.47]	11.149 [9.937,12.36]

Table 12: Number of Hops Inaccuracy Level under Different Traffic Rates and Using Default OLSR Parameters

It can easily be noticed that under all traffic rates and using the default OLSR parameters, the farther the nodes are (the higher the distance between the nodes) the less accurate the state information is. This phenomenon is caused directly by two factors. First, the nature of the energy level as a monotonically decreasing metric and second, the message propagation time, which is a function of the distance (number of hops) since a message takes longer time to reach farther nodes.

Having this trend observed, and keeping in mind the goal of whether it is possible to improve state information accuracy, the impact of varying the OLSR parameters on state information accuracy at different hops is further analyzed. The question under investigation is how accuracy for farther apart nodes can be improved. OLSR parameters might help in this matter in two ways. First, increasing the frequency of update messages supplies more accurate information and provides faster recovery in case of message delay and loss. Second, advertising more information ensures the availability of more up-to-date information throughout the network.

Figure 16 shows the number of hops inaccuracy level under the different traffic rates and using different OLSR parameters.

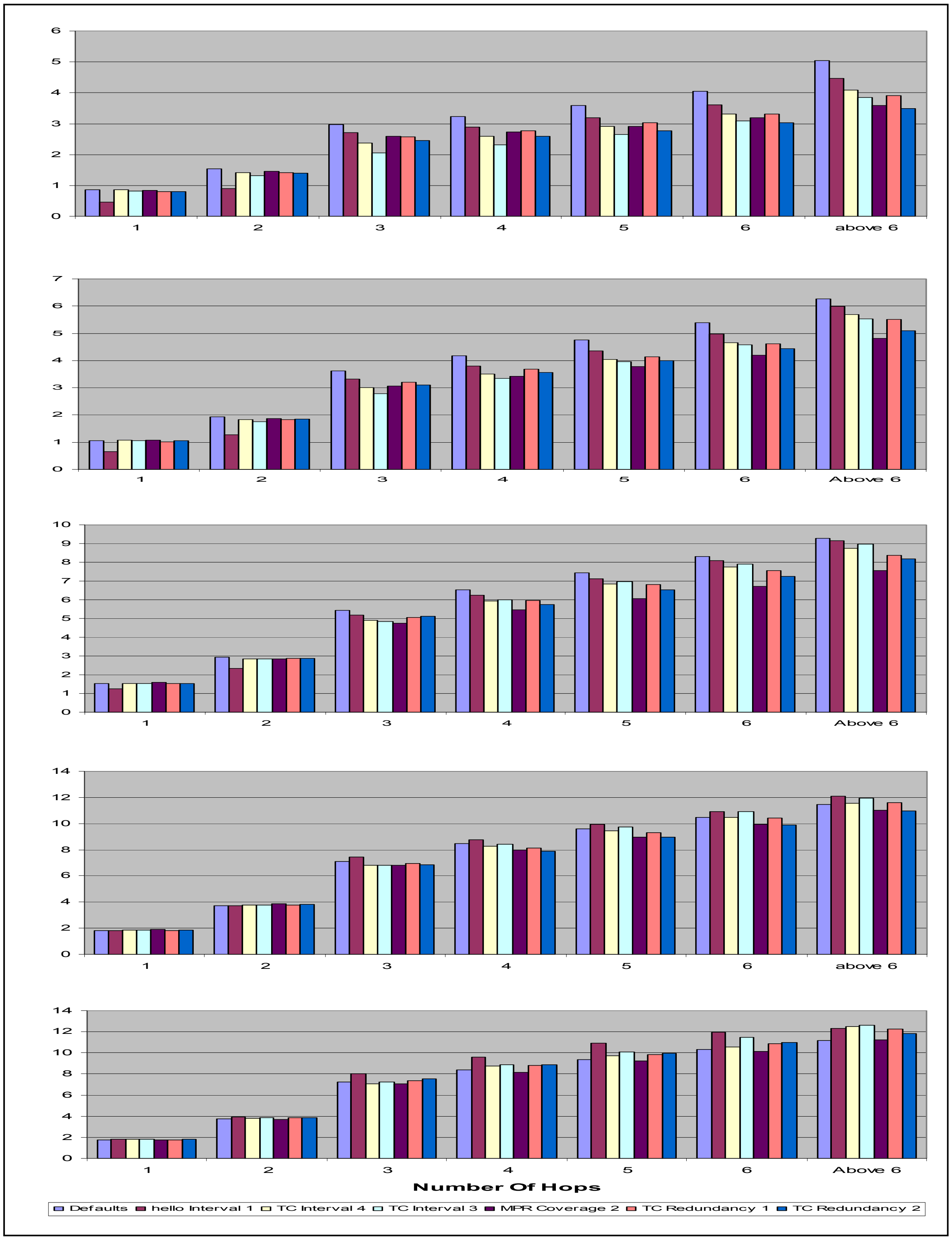


Figure 16: The Impact of Varying OLSR Parameters on Number of Hops Inaccuracy Level under 0.2, 0.14, 0.09, 0.04 and 0.02 Traffic Intervals Respectively

Following are the main observations for Figure 16. All these observations are based on a 95% confidence interval test:

- Under the low traffic rate (0.2 traffic interval), nodes obtained more accurate energy level information about all other nodes (regardless of the distance) when varying the OLSR parameters.
    - Using a Hello-interval of 1 achieved significant improvements in inaccuracy level for nodes that are 1 and 2 hops away compared to farther nodes since it provides more frequent information for 1 and 2 hop neighbors. And then TC messages are loaded with more recent information leading to more accurate information throughout the network.
    - For all the other OLSR parameters (TC-interval 3 and 4, MPR-coverage 2 and TC-redundancy 1 and 2), the improvements in inaccuracy levels were significant for farther nodes (nodes at hop 3 and higher). All those parameters impact the TC messages by either increasing the number of messages or advertising more redundant information. And since it is by exchanging TC messages that a node discovers farther away nodes (at hop 3 and higher), hence more accurate information at higher hops matches our expectations.
- Under medium traffic rate (0.14 and 0.09 traffic intervals), in general there is a trend towards more accurate information.
    - More accurate information about nodes that are 1 and 2 hops away is achieved only by increasing the frequency of Hello messages.
    - Knowledge about energy levels about more distant nodes is improved by varying the OLSR parameters especially TC-interval 3, MPR-coverage 2 and TC-redundancy 2 where a 95% CI test suggested that the improvement is statistically significant.

- Under high traffic rate (0.04 and 0.02 traffic intervals), varying the OLSR parameters did not have a positive impact on the overall inaccuracy level. On the contrary, the trend was towards less accurate information especially under traffic interval of 0.02 and for farther nodes. This is due to the elevated levels of message loss and delay as a result of network congestion.

As a result:

- Hello-interval 1 is the only parameter that improves the accuracy of energy level information for 1 and 2 hop neighbors under low and medium traffic rates. Therefore any set of parameters to be used to improve inaccuracy level should include Hello-interval 1.

- It is the responsibility of TC-interval 3, TC-interval 4, MPR-coverage 2, TC-redundancy 1 and TC-redundancy 2 to improve the accuracy of energy level information for farther nodes.

- Having the Hello-interval set to 1, it makes sense to decrease the TC-interval to 3 since according to the OLSR specification, the ratio between Hello-interval to TC-interval is 2 to 5.

- Setting the TC-interval to 3 matches well with our results. According to the CI test and comparing the different OLSR parameters, it shows that TC-interval 3 starts off outperforming the other parameters under lower traffic rates and as traffic rate increases, MPR-coverage 2 does better for more distant nodes (5 hops and higher). Under higher traffic rates (0.02 traffic interval), the difference between TC-interval 3 and MPR-coverage 2 is considered to be not statistically significant.

  To decide between TC-interval 3 and MPR-coverage 2, we have looked at the overhead associated with these two parameters. The overhead is calculated as the percentage of total number of TC messages sent using each parameter individually, compared to using the defaults parameters.

It is evident from Table 13 that MPR-coverage of 2 imposes a high overhead on the network under low to medium traffic compared to TC-interval of 3. Added to the fact that none of these parameters seem to improve inaccuracy under high traffic rates, a TC-interval 3 will be used as an attempt to reduce inaccuracies.

Traffic Interval	Overhead Using TC-interval 3	Overhead Using MPR-coverage 2
0.2	65%	123%
0.14	65%	121%
0.09	58%	108%
0.04	53%	85%
0.02	58%	80%

Table 13: The Overhead Using TC-Interval 3 compared to MPR-Coverage 2

5.5.4. Overall Inaccuracy Level under Hello1Tc3 OLSR

Based on the energy level number of hops inaccuracy level analysis described in the previous section (5.5.3), the results show that it might be promising to use a combination of Hello-interval 1 and TC-interval 3 as OLSR parameters.

Figure 17 compares the overall inaccuracy level using the default parameters for OLSR ,called default OLSR, versus using a combination of Hello-interval 1 and TC-interval 3 and keeping the other parameters unchanged (MPR-coverage 1 and TC-redundancy 0), called Hello1TC3 OLSR.

Hello1TC3 OLSR improves the overall inaccuracy level under low traffic rates and as we move towards high traffic rates, Default OLSR starts to outperform Hello1TC3 OLSR due to the overhead added to the network.

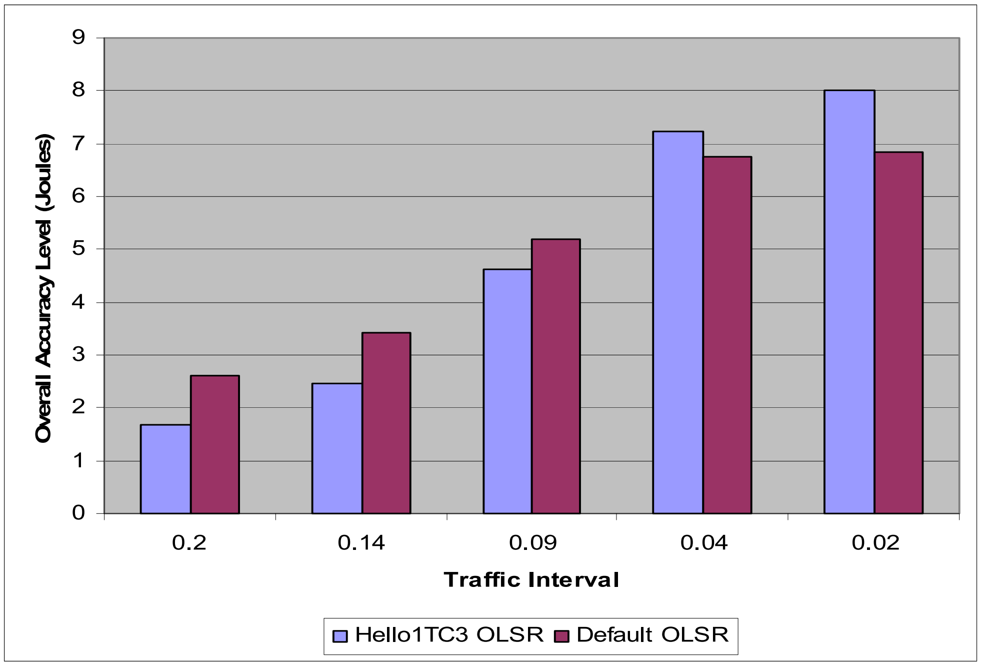


Figure 17: Overall Inaccuracy Level of Default OLSR vs. Hello1TC3 OLSR

Traffic Interval	Overall Inaccuracy Level Default OLSR	Overall Inaccuracy Level Hello1TC3
0.2	2.6225 [2.37,2.87]	1.6887 [1.49,1.89]
0.14	3.4287 [2.96,3.9]	2.461 [2,2.92]
0.09	5.1793 [4.8,5.56]	4.6302 [4.05,5.21]
0.04	6.7602 [5.94,7.58]	7.231 [6.42,8.04]
0.02	6.839 [5.86,7.81]	8.003 [6.6,9.41]

Table 14: Overall Inaccuracy Level under Default OLSR Parameters and Under a Combination of Hello-Interval 1 and TC-Interval 3

Since the performance of any routing protocol is determined by Packet Delivery Ratio and Per Packet Delay, Figures 18 and 19 compare packet delivery ratio and packet delay (respectively) for Default OLSR and Hello1TC3 OLSR.

- Packet Delivery Ratio: the percentage of packets that successfully reach the destination nodes. Packet Delivery Ratio = (packets received / total packets sent) * 100%.

- Per Packet Delay (PPD): the average time between a packet being sent and being received.

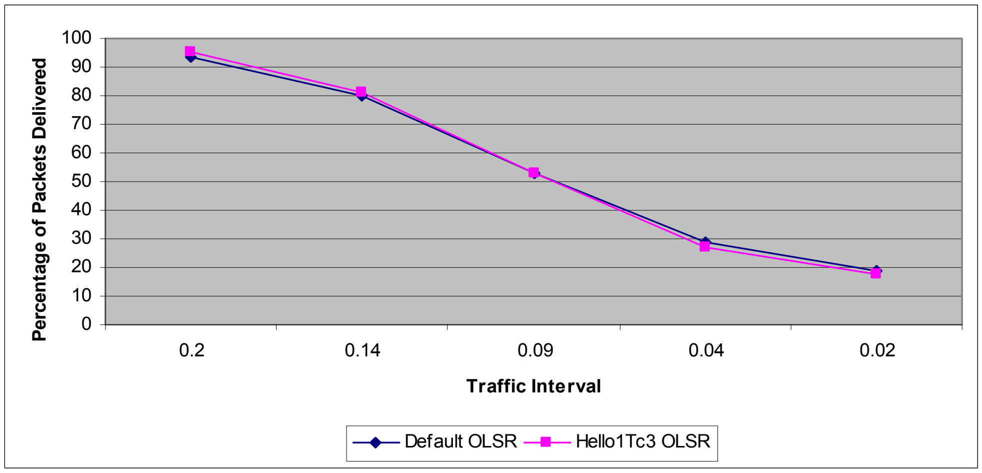


Figure 18: Comparison between Packet Delivery Ratio under Hello1TC3 OLSR versus Default OLSR

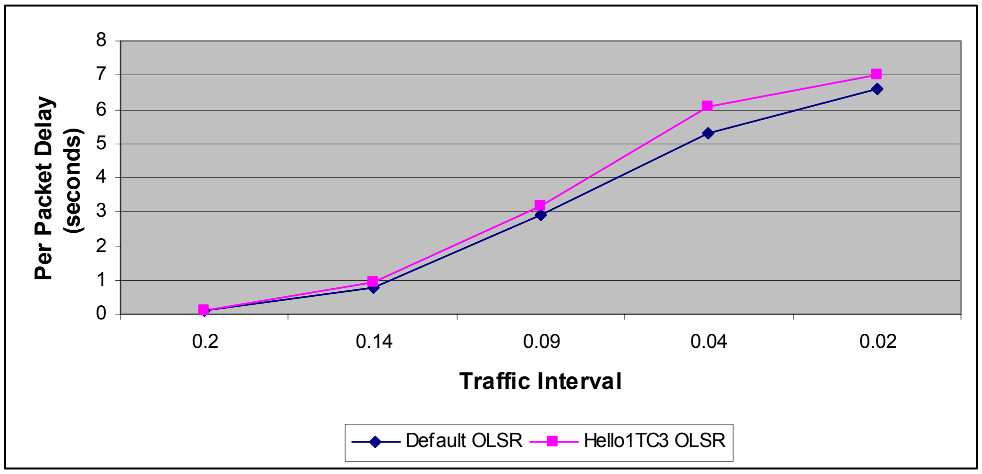


Figure 19: Comparison between Per Packet Delay under Hello1TC3 OLSR versus Default OLSR

As shown in Figure 18, the packet delivery ratio is very similar under Default OLSR and Hello1TC3 OLSR. But according to Figure 19, sending extra protocol messages increases the PPD for higher traffic rates.

In a summary, under high traffic rates, any added overhead will degrade the network performance in terms of both less accurate information as well as higher delays. On the other hand, under lower traffic rates, sending extra protocol messages provided more accurate view of the network while maintaining the same delivery rates and acceptable delays.

5.5.5. Knowledge Age Inaccuracy Level

Knowledge age inaccuracy level represents the average difference between a node's actual energy level and what other nodes believe its energy level is, categorized by how old the knowledge is. It is calculated exactly the same way it is done for the queue length metric in section 5.4.4.

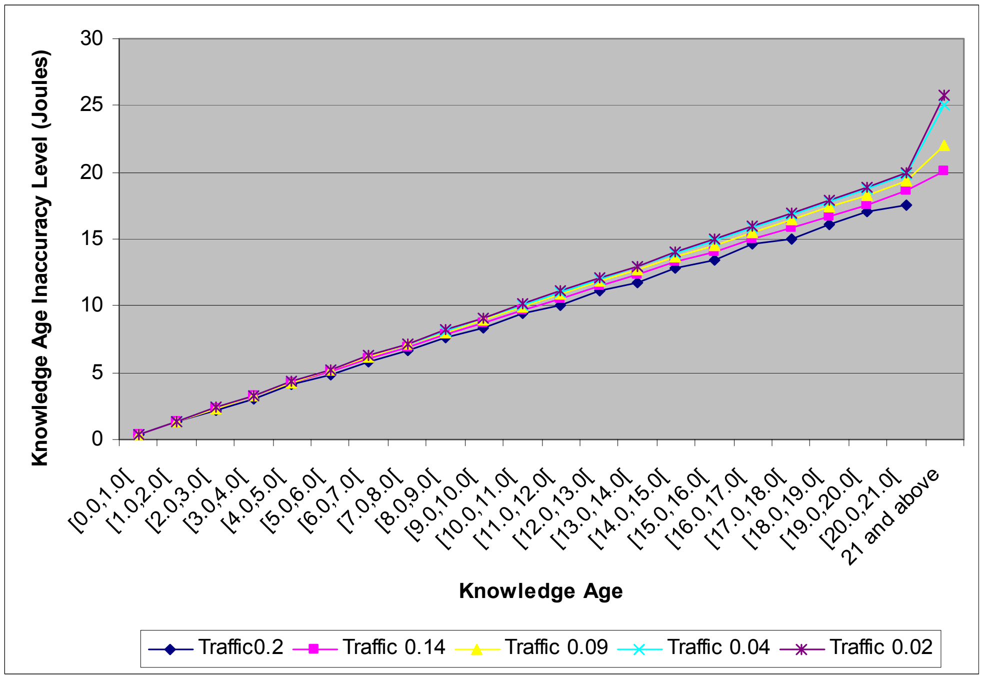


Figure 20: Knowledge Age Inaccuracy Level under Different Traffic Rates and using Default OLSR Parameters

It can be clearly seen from Figure 20 that the older the knowledge about other nodes is, the less accurate their energy levels are. This phenomenon is inherited from the characteristics of the energy level as a monotonically decreasing metric. Based on these observations, we will explore ways to increase energy level accuracy in Chapter 7.

5.6. Conclusion

The experiments have shown that there is considerable level of state information inaccuracy for both the queue length and energy level metrics. The characteristics of the OLSR protocol, the underlying topology, and traffic load all contribute to the high levels of inaccuracy. As a matter of fact, it will be shown later in Section 6.5 that inaccuracy level under higher traffic rates is close to the level we would obtain if nodes randomly guessed the queue lengths of other nodes. Obviously, this then defeats the purpose of propagating the state information.

Tuning the OLSR protocol parameters did not seem to positively impact the overall inaccuracy level for both metrics. To get better insights of what can be done to improve inaccuracies, analysis based on number of hops and knowledge age was conducted for both the queue length and energy level. The number of hops analysis showed that, under low traffic rate, a little bit of additional accuracy in energy level information can be obtained by using a combination of Hello-interval 1 and TC-interval 3 for OLSR parameters. The same behavior was not obtained either under high traffic rates or for the queue length metric. The knowledge age analysis for both queue length and energy level metrics showed that inaccuracy level is highly affected by the age of the data, and suggested that any solution to reduce inaccuracies has to deal with the outdated knowledge.

Based on our observations, Chapters 6 and 7 will explore ways to overcome the old knowledge issue for the queue length and energy level respectively. New techniques will be proposed and evaluated against the basic OLSR.

Chapter 6

QUEUE LENGTH INACCURACY IMPROVEMENT

6.1. Introduction

Section 5.4.1 analyzed the queue length overall inaccuracy level under five different traffic rates using different OLSR parameters. Tuning the OLSR protocol parameters did not seem to have a positive impact on inaccuracy level. Analysis based on number of hops and knowledge age was conducted to get better insights of what can be done to improve inaccuracies.

Number of hops and knowledge age analysis concluded that the inaccuracies are mostly caused by nodes that are far away or equivalently nodes that we have only "old" knowledge for.

In this chapter, we will look at methods to improve queue length inaccuracy level. We propose two techniques: Probing (Section 6.3) and Exponential Averaging (Section 6.6). Then we compare the proposed techniques to the Default OLSR solution (OLSR with the QoS-related state propagated and using the default parameters).

6.2. Threshold-Based Updates

Based on the number of hops and knowledge age analysis described in Sections 5.4.4 and 5.4.5 respectively, the results show that inaccuracy level is highly affected by the age of the data especially under higher traffic rates. It is also affected by how far away the node is. We also showed that there is a direct correlation between how many hops away the node is and how old the knowledge about it is. The conclusion was to propose a strategy that takes care of either one of the two factors and the solution in such situations is threshold-based updates.

Threshold-based update is a policy in which an update message is generated based on a trigger. The policy is characterized by a threshold value, and to control the overhead, a hold-down timer specifies the minimum time intervals between consecutive updates of the same piece of information.

The threshold value and hold-down timer are model based parameters. In other words, what these parameters present, and their values depend on the objective and the design of the update strategy. A threshold value might present a QoS-related state change. If a node finds there is a "significant QoS-related state change", it will generate a message to announce its new value and enable other nodes to update their knowledge to reflect such changes. Also, a threshold value might present a time interval in which:

- At receiving node: If no update was received from a node during that period of time, the data becomes obsolete and the node is probed for more recent information.
- At transmitting node: If a node suspects that other nodes might have old information about it, the node sends an update message informing the other nodes in the network about the change.

In either case the threshold value must be set properly such that it meets the objective of the update strategy while maintaining a reasonable overhead. If the threshold value is low, that will cause frequent generation of update messages, introducing high overhead, although theoretically more accurate information is obtained. On the other hand, if the threshold value is high, overhead is reduced at the expense of relatively less accurate information. In addition, since rapid fluctuations in the QoS metric or long delays and high loss rates due to network congestion can generate a large number of state information updates, a hold-down timer is used to control the overheads. It specifies a time interval in which a node is limited to a maximum of one probe message generation. There is no limit on the number of probe messages relayed.

For the purpose of eliminating or reducing the existence of outdated "old" knowledge, a threshold value that presents a time interval seems to be appropriate, in particular at the receiving node.

The problem with implementing the update policy at the transmitting node is that it is hard for a node to decide if other nodes have old information about it. According to OLSR, nodes send periodic updates; a Hello message every 2 seconds and a TC message every 5 seconds, but they never know if the message is lost or delayed. Also to make such an update available to all nodes in the network, a new message type has to be implemented since Hello messages reach only 2-hop neighborhood and TC messages are sent by MPR nodes only. The new message could be either a broadcast or can be sent by any node but forwarded by MPRs only to control overhead. But since a node does not have an assurance that other nodes have old knowledge about it, it might send more messages than required, leading to higher overheads or less messages than needed and consequently less accurate information.

We adopted the option of implementing the update policy at the receiving node. Whenever a node has old knowledge about another node in the network, it sends a probe message to that node requesting its most recent information. We called our proposed strategy *Probing* Technique.

6.3. Probing Technique

In designing a Probing technique, the main issues to be looked at are:

- When to generate the probe message
- What is the structure of such a message
- Upon receiving a probe message, how the message is processed and
- How often a probe message is generated.

As mentioned in Section 6.2, this type of update policy is based on a trigger and is characterized by a threshold value and a hold-down timer. And these parameters are model specific. In our model:

- Trigger: upon receiving a protocol message (hello or TC message), a node updates its knowledge and if it has information about another node that is beyond the threshold value, it runs the probe message generation routine. If a node has old knowledge about more that one node, it selects the node with the oldest information.
- Threshold value *th*: the performance of the probing technique is studied under different threshold values.
- Hold-down timer: set to 2 seconds, therefore, a node sends a maximum of 1 probe message every 2 seconds.

6.3.1. Probe Message Structure

A probe message should specify the "final destination" which is the address of the node to be probed. And since we want to minimize the overhead by limiting the number of retransmissions, only the nodes along the path to the final destination are responsible for retransmitting the message. Therefore, a probe message should also specify the "next hop" which is the address of the node responsible for retransmitting the message. Also, a "flag" is used to mark whether the message is a probe-request or a probe-reply since in our implementation we use only one message type. And for statistical analysis, a "timestamp" of when the message was originally generated is added to the probe message.

In addition to that, as an attempt to make use of the probe message at intermediate nodes, before a node sends out a probe message (either generate or retransmit), it loads the message with all the knowledge it has about other nodes in the network. This includes which nodes it can reach, along with their queue lengths and associated timestamps, similar to Hello messages and TC messages. Therefore, as the probe message is being propagated through the network, nodes collect information and update their knowledge and also a transmitting node updates the entries of the message with the most recent information. It should be noted that since probe messages are broadcast, even nodes that are not on a path may learn new information from a probe message by promiscuous listening.

The structure of the probe message is shown in Figure 21. It is worth mentioning that similar to Hello and TC message, this is sent as the data-portion of OLSR General Packet format with the "Message Type" set to PROBE_MESSAGE.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       final destination                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       next hop address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       request/reply flag                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Address                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        QoS metric value                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                               ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Probe Message Structure

6.3.2. Probe Message Generation and Processing

Upon receiving a Hello or TC message, a node collects the queue length information about other nodes and updates its knowledge. If it finds out there is a node with queue length information that is more than *th* seconds old, it runs the probe message generation routine. However, a node does not generate a probe message unless it did not send a probe message for the last 2 seconds (the value of the hold-down timer)

For example let us assume that node *A* want to probe node *D* as shown in Figure 22. It first determines the next-hop node on its route to node *D*. It then generates a probe message with the final destination equals to "*D*", next hop equals to "*B*" and the flag set to "*request*". It also loads the message with the most recent information it (node *A*) has about all the nodes in the network.

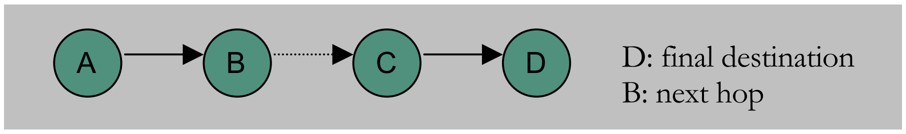


Figure 22: Node A Wants to Send a Probe Message to Node D

Upon receiving a probe message, a node processes the message based on the probe message processing algorithm.

- If a node receives a probe-request that is destined to it, it will reply with a probe-reply message, which is propagated back to the originator node.

- If a node receives a probe-reply message that is destined to it, it updates its knowledge and then discards the message.

- Whenever a node receives a probe message and finds out it is the designated node to relay the message, it updates the message and similarly relays it.

- If a node that is neither the final destination nor the next-hop node receives a probe message, it updates its knowledge and then discards it.

The details of the algorithm are given below:

```
if the message is probe-request
  if node address == final destination address
    //Send a reply to the originator of the message
    final destination = originator of the message
    next hop = address of first node along the path from this node to originator address
    set the request/reply flag to "reply"
    update the message entries with the most recent queue length information and same for the node,
    update its knowledge with the most recent information
    generate probe-reply message

  else if node address == next hop address
    //relay the message
    final destination = final destination
    next hop = address of first node along the path from this node to final destination
    keep the request/reply flag to "request"
    update the message entries with the most recent queue length information and same for the node,
    update its knowledge with the most recent information
    relay the message

  else
    //just process the message
    update the queue length with the most recent information

else  //the message is probe-reply
  if node address == final destination address
    //the reply came back to the originator of the message, process it and stop here
    update the queue length information

  else if node address == next hop address
    // relay the message
    final destination = final destination
    next hop = address of first node along the path from this node to final destination
    keep the request/reply flag to "reply"
    update the message entries with the most recent queue length information and same for the node,
    update its knowledge with the most recent information
    relay the message

  else
     //just process the message
    update the queue length with the most recent information
```

6.4. Overall Inaccuracy Level under Probing

During the simulation, in addition to the data collected by the data collection module described in Section 5.3, we keep track of:

- All probe messages sent, including both generated and relayed messages for overhead calculations.
- All probe messages generated (that is excluding retransmissions). This is particularly helpful when choosing a threshold value.
- Total number of probe messages received.
- Total number of times we receive a reply to a probe-request, including when the request was generated and when the reply was received. A round-trip delay is calculated as the average difference between the time the request is sent out and when the reply is received.
- When reporting the perceived values, how many times a probe message supplied more recent information than a node has, and how much more recent this information is.

6.4.1. Threshold Value

The only issue we have to look at before proceeding to analyzing the performance of the probing technique is the threshold value. As mentioned early on in this chapter, the threshold value is a very critical parameter in such situations. Since old knowledge is believed to be the main source of inaccuracy, a tradeoff between how much old knowledge is allowed and how many probe messages are sent has to be made.

Figure 23 illustrates the relation between knowledge age and number of occurrences. Number of occurrences refers to how many pairs of nodes (n1, n2) exist such that n1 has knowledge about n2 that is in the range [x, x+1[seconds old and x = 0 to 21.

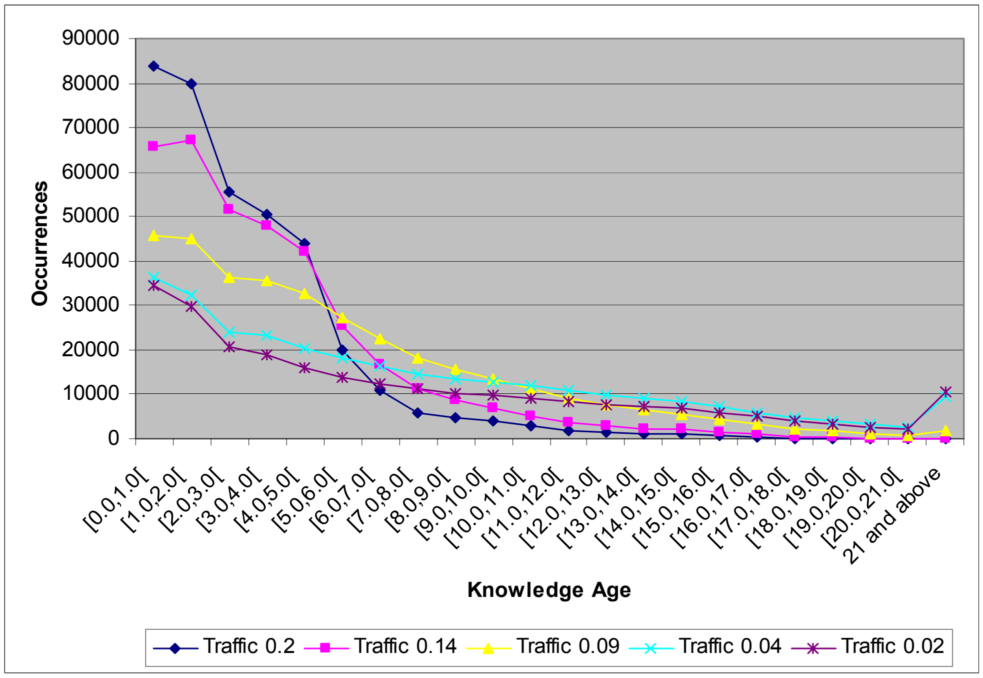


Figure 23: Knowledge Age vs. Occurrences under Different Traffic Rates and using Default OLSR Parameters

The first threshold value we experimented with is *th* = 7. There are mainly 2 reasons for such a selection. First as shown in Figure 23 above, the number of occurrences curve starts to flatten around this value. And second, according to the results in Chapter 5, we have an average knowledge age of about 7 seconds (under high traffic rate).

In the next section, we will study the inaccuracy levels under the probing technique. First we will use a threshold value of 7. Whenever a node has knowledge about another node that is older than 7 seconds, it runs the probe message generation routine. We will look at different performance aspects to improve the inaccuracy level under our proposed probing technique.

6.4.2. Performance Measurements of the Probing Technique

The simulation results presented are an average over 10 scenarios and the same set of 10 scenarios throughout this research is used to ensure fair comparison under the same conditions.

Threshold Value *th* = 7:

Traffic Interval	Overall Inaccuracy Level under Default OLSR	Overall Inaccuracy Level under Probing *th* = 7
0.2	0.17 [0.05, 0.29]	0.235 [0.15,0.32]
0.14	1.49 [0.79, 2.91]	1.522 [0.9,2.15]
0.09	4.88 [3.66, 6.1]	4.44 [3.34,5.54]
0.04	8.48 [6.86, 10.09]	7.62 [6.48,8.75]
0.02	9.39 [7.72, 11.06]	9.76 [8.34,11.18]

Table 15: Overall Inaccuracy Level under Probing with th = 7 vs. Default OLSR

As shown in Table 15, according to a 95% confidence interval test, under different traffic rates the difference between the overall inaccuracy levels under probing with *th* = 7 versus Default OLSR is considered to be not statistically significant.

Table 16 shows the percentage of the total number of times a probe message provides more recent information about a node and also how much more recent this information is. The results show that there is a considerable number of knowledge updates (number of times a node learns more recent information about another node) from probe messages, but how effective are these updates?

Traffic Interval	Percentage of Knowledge Updates from Probe Messages	Average Difference (old → new)
0.2	36%	4.8 → 2.13
0.14	42%	5.63 → 2.66
0.09	54%	7.3 → 3.63
0.04	30%	9.7 → 5.47
0.02	15%	11.11 → 6.5

Table 16: Percentage of Times a Node Learns More Recent Information about another Node from a Probe Message

As a matter of fact, the more important piece of information is the average difference, which is by how much these updates are more recent. Figure 12 describes the relation between knowledge age and inaccuracy level. Although probe messages provide more recent information, it is not enough to improve inaccuracy levels since knowledge age is still in the same range of inaccuracy levels. For example, under a traffic interval of 0.14, probe messages provided more recent information 42% of the time. On average, the update was from data that is 5.63 seconds old to more recent data that is only 2.66 seconds old. Based on Figure 12, inaccuracy level associated with 5.63 knowledge age is about 2, which is the same as its counterpart of 2.66 seconds of knowledge age.

The problem here is that even with the probe messages providing more recent information, this information is still considerably old, causing high levels of inaccuracy. There are some factors that might be contributing to the high levels of inaccuracy in addition to the other factors discussed in the previous chapter. The most important one is the overhead resulting from the extra probe messages sent.

Table 17 shows the overhead which is represented as the total number of probe messages sent (generated or relayed) in 10 simulations. It is evident that probe messages are generated at a very high rate. For example under a traffic interval of 0.14, there are about 62,000 messages sent during 10 simulations of 200 seconds each. Therefore, on average of 30 probe messages are sent per second. We also compare the total number of probe messages sent to the total number of TC messages sent, and it can be seen clearly that probe message are sent at a very high rate.

It is important to point out that according to Table 17 as traffic rate increases, number of probe messages sent increases. This is due to the fact that as nodes send more messages, the network becomes congested, causing higher loss rate and larger delays. As a result knowledge about other nodes in the network is not updated in a timely fashion, triggering the generation of probe messages. On the other hand, under high traffic rates, nodes get disconnected and routes from a source to a destination become unavailable. Therefore, lots of probe messages cannot be relayed and dropped.

Traffic Interval	Overhead	Percentage compared to TC messages sent
0.2	55214	30%
0.14	61862	35%
0.09	77413	50%
0.04	39395	34%
0.02	21509	20%

Table 17: Probing Technique with th = 7 Overhead

The previous results concluded that a threshold value of 7 is very low. The overhead was very high to a degree that even the knowledge updates from probe messages were not effective.

The objective here is to reach a threshold value where very few probe messages are sent per second, especially under high traffic rates. At the same time we also need to examine the performance of our probing technique under the different threshold values.

Threshold Value *th* = 17

Table 18 shows the overall inaccuracy level of the probing technique using a threshold value of 17 versus default OLSR. A 95% confidence interval test suggests that the difference between the two overall inaccuracy levels is considered to be not statistically significant. Moreover, it can be easily noticed that the overhead is still high, especially as traffic rate increases, when on average 6 to11 probe messages are sent every second.

Traffic Interval	Overall Inaccuracy Level under Default OLSR	Overall Inaccuracy Level under Probing *th* = 17	Overhead
0.2	0.17 [0.05, 0.29]	0.167 [0.09,0.24]	6208
0.14	1.49 [0.79, 2.91]	1.49 [0.75,2.24]	8150
0.09	4.88 [3.66, 6.1]	4.56 [3.39,5.72]	21973
0.04	8.48 [6.86, 10.09]	7.9 [6.35,9.46]	19876
0.02	9.39 [7.72, 11.06]	9.67 [8.08,11.26]	12500

Table 18: Overall Inaccuracy Level under Probing with th = 17 vs. Default OLSR and Associated Overhead with the Probing Technique

Threshold Value *th* = 21

Table 19 shows the overall inaccuracy level of the probing technique using a threshold value of 21 versus default OLSR. Unfortunately, the difference between the two overall inaccuracy levels is considered to be not statistically significant according to a 95% confidence interval test although, at this threshold value, the number of probe messages sent has dropped to acceptable levels. As a matter of fact, probe messages are rarely sent at low traffic rates, while under medium to high traffic rates on average 4 to 6 probe messages are sent per second.

Traffic Interval	Overall Inaccuracy Level under Default OLSR	Overall Inaccuracy Level under Probing *th* = 21	Overhead
0.2	0.17 [0.05, 0.29]	0.124 [0.08,0.17]	123
0.14	1.49 [0.79, 2.91]	1.38 [0.74,2.02]	756
0.09	4.88 [3.66, 6.1]	4.94 [3.81,6.06]	9388
0.04	8.48 [6.86, 10.09]	8.13 [6.59,9.67]	12360
0.02	9.39 [7.72, 11.06]	9.43 [8.03,10.83]	7982

Table 19: Overall Inaccuracy Level under Probing with th = 21 vs. Default OLSR and Associated Overhead with the Probing Technique

To be able to conclude the analysis of the probing technique performance, one more attempt with a higher threshold value is undertaken.

Threshold Value *th* = 25

Similar to the previous threshold values, there is no improvement in inaccuracy level under the probing technique as shown in Table 20 for a threshold value of 25. Under a low traffic rate of 5 packets per second (0.2 traffic interval) no probe messages are sent, meaning that nodes do not have data about other nodes that is older than 25 seconds. On the other hand, under higher traffic rates, on average 2 to 5 probe messages are sent.

Traffic Interval	Overall Inaccuracy Level under Default OLSR	Overall Inaccuracy Level under Probing *th* = 25	Overhead
0.2	0.17 [0.05, 0.29]	0.124 [0.08,0.17]	0
0.14	1.49 [0.79, 2.91]	1.478 [0.76,2.19]	43
0.09	4.88 [3.66, 6.1]	4.8 [3.73,5.87]	4105
0.04	8.48 [6.86, 10.09]	8.21 [6.77,9.65]	9872
0.02	9.39 [7.72, 11.06]	9.3 [7.79,10.8]	7596

Table 20: Overall Inaccuracy Level under Probing with th = 25 vs. Default OLSR and Associated Overhead with the Probing Technique

6.4.3. Summary of the Probing Technique with the Different Threshold Values and Default OLSR

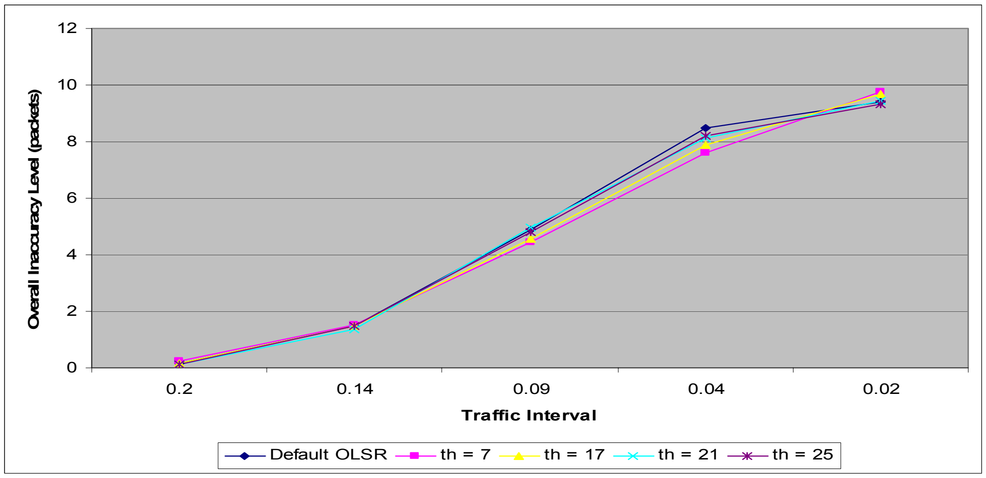


Figure 24: Comparison between the Probing Technique with the Different Threshold Values and Default OLSR

As illustrated in Figure 24 above, probing did not seem to positively impact the overall inaccuracy level under the different threshold values of 7, 17, 21 and 25. Under all threshold values, there is a large number of knowledge updates from probe messages. But these updates move the knowledge age within a range where no improvements in inaccuracy are obtained.

A threshold value of 21 seems to be the most appropriate since it provides quite a large number of knowledge updates from probing while keeping the overhead at an acceptable level. Therefore, for the remaining part of our discussion on the probing technique, a threshold value of 21 is assumed.

We next investigate some factors that might have contributed to the unexpected behavior of the probing technique. Table 21 shows the number of probe-requests generated excluding retransmissions compared to the number of replies (to requesting node) received.

Traffic Interval	Request to Reply Ratio
0.2	3/38
0.14	11/334
0.09	80/4063
0.04	120/6411
0.02	87/4445

Table 21: Probe-Request to Probe-Reply Ratio

The small number of replies indicates a high probe messages loss rate and delay. Other than the common factors for any network as mentioned in the previous chapter, following are some of the reasons that are specific to our probing technique:

- Only one node is responsible for relaying a probe message. Consequently, a high percentage of messages are lost or delayed due to broken links or buffer overflow.

- When a node generates a probe message, it loads it with all the information it has about other nodes in the network. And as the message is being relayed, every intermediate node updates the message (or add to it if not included) with the most recent information. Consequently, probe messages are quite large in size. This might cause higher loss rates and larger delays.

Table 22 shows the overall inaccuracy level under a modified version of our probing technique using the threshold value of 21. In the *Modified Probing Technique*, instead of loading the message with all the information a node has, every node puts only its own information (address, queue length and time) and as the message is being relayed, only information about the relaying node itself is added to the message. This guarantees that recent information about active nodes only is included in the message.

Traffic Interval	Overall Inaccuracy Level under Default OLSR	Overall Inaccuracy Level under Modified Probing *th* = 21
0.2	0.17 [0.05, 0.29]	0.124 [0.08,0.17]
0.14	1.49 [0.79, 2.91]	1.46 [0.75,2.17]
0.09	4.88 [3.66, 6.1]	4.92 [3.63,6.21]
0.04	8.48 [6.86, 10.09]	8.43 [7.01,9.85]
0.02	9.39 [7.72, 11.06]	9.52 [7.84,11.2]

Table 22: Overall Inaccuracy Level under Probing with th = 21 and Reduced Probe Message Size vs. Default OLSR

Reducing the size of the probe messages did not improve the inaccuracy level. But we observed that a higher number of probe messages is sent. This might be due to lower delays and loss rates since nodes have limited processing power.

Probing is potentially an efficient method in maintaining more accurate information, since state information is updated whenever the threshold value is passed without waiting for the next periodic state information update. However probing did not increase accuracy for the queue length as our metric. *Is that the best inaccuracy level that can be obtained for this type of QoS metric?*

6.5. Random Guessing versus Queue Length Propagation

Is it worthwhile propagating the queue length information throughout the network, because as shown especially under medium to high traffic, the inaccuracy level is quite high? Will propagating the queue length information be better than randomly guessing the queue length of other nodes?

To answer the question of what is the inaccuracy level if each node randomly guessed the queue length value for other nodes, we laid out a simple guessing model. This model is based on two assumptions. First, queue length is uniformly distributed in [0, 49] and second, nodes guess based on a uniform distribution. Given below is the pseudo code:

```
begin
  MAXQUEUELENGTH := 50
  int sum = 0;
  for (i = 0; i < MAXQUEUELENGTH; i++)
  begin
    for (j = 0; j < MAXQUEUELENGTH; j++)
    begin
        sum += random value between 0 and 49 inclusive;
    end
  end
  sum = sum/(MAXQUEUELENGTH * MAXQUEUELENGTH);
end
```

Under this simple guessing model, the result shows an average inaccuracy level of 16.6 which is higher than the inaccuracy level observed with default OLSR under all traffic loads. But is it possible to design a smarter model that takes into account the traffic load and the actual queue length distribution and try to estimate a more accurate queue length?

Figure 25 describes the queue length change under all traffic rates in a 1 second interval. Queue length change refers to the percentage of time where over a specified interval, the difference between the highest and lowest queue length for a node is less than or equal to x and $x = 0$ to 49. Figure 25 shows that 50% of time nodes change their queue length in a 1 second interval by 1 to 4 (low to high traffic rate). The rate of queue length change could be used to guess/estimate a new queue length value based on an old reported value.

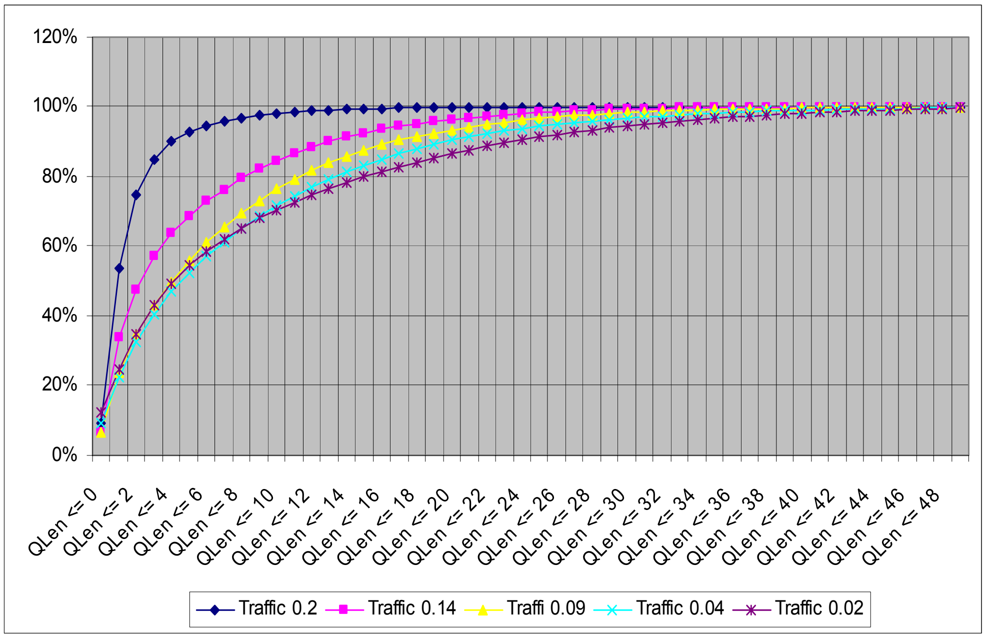


Figure 25: Queue Length Change in 1 sec Interval under Different Traffic rates and using Default OLSR Parameters

But, if we know for example that under high traffic rates, the average knowledge age is 7, can we analyze the actual queue length distribution in 7 seconds interval and try to make a better guess for nodes' queue lengths. As part of future work, can we design a model such that a node needs to associate some stochastic behavior like an experimentally derived *probability distribution function* to the queue length/queue length change and extrapolates a more accurate value?

6.6. Exponential Averaging

Whenever a node receives more recent queue length information about another node, it believes that this is the new value and all the analysis is done based on this new value. Knowing that the queue length is a very dynamic metric, can we improve the performance by taking into account the old queue length value.

We designed a model based on exponential averaging where instead of setting the queue length to be the new value; we use a combination of the old and new value. Exponential averages are used to smooth out short-term fluctuations, thus highlighting longer-term trends:

$$\text{Expected queue length} = \alpha * (\text{new value}) + (1 - \alpha) * (\text{old value})$$

where α is a parameter to balance between short-term and long-term data. It is in the range [0, 1]. For example if the old queue length = 10 and the new queue length = 20 and we want the expected value to be based 80% on the new value and 20% on the old value, we set α to 0.8. Hence, the expected value = 0.8*20 + 0.2*10 which equals to 18.

There is always a difficulty when analyzing a model involving a threshold value, since it is very crucial to decide on which threshold value to use. We ran experiments with threshold values of 0.9, 0.8, 0.7, 0.6, 0.5 and 0.1. Table 23 shows the associated overall inaccuracy levels under all traffic rates.

Traffic Interval	90-10%	80-20%	70-30%	60-40%	50-50%	10-90%
0.2	0.171	0.171	0.171	0.172	0.173	0.18
0.14	1.49	1.49	1.5	1.52	1.54	1.71
0.09	4.86	4.87	4.91	4.97	5.04	5.51
0.04	8.45	8.48	8.53	8.6	8.7	9.31
0.02	9.37	9.38	9.42	9.48	9.57	10.12

Table 23: Overall Inaccuracy Levels under Exponential Averaging using threshold values of 0.9, 0.8, 0.7, 0.6, 0.5 and 0.1 under all Traffic Rates

As shown in the table above, the overall inaccuracy levels are about the same under the different threshold values. But the trend is towards less accurate queue length information as we base our expected values more on the old values. This trend was confirmed when a threshold of 0.1 is used.

6.7. Conclusion

Two techniques were proposed to reduce queue length inaccuracies. Both the probing and exponential averaging techniques did not positively impact the inaccuracy level. Under the probing technique, whenever a node has old knowledge about another node, it sends a probe message to that node requesting its most recent information. The probing technique was evaluated under different threshold values for knowledge age. Although there was a large number of knowledge updates from probe messages, these updates moved the knowledge age within a range where no improvements in inaccuracy level were obtained. It was concluded that messages loss and delay as a result of increased overhead might have contributed to the unexpected behavior.

Similarly the exponential averaging technique was evaluated under different threshold values. Threshold values are used to smooth out the short term fluctuations and thus highlighting long term trends. The results have shown no improvements in inaccuracy levels under all traffic rates and all threshold values. In contrast, the trend was towards less accurate queue length information when aggressively "smoothing out" recent values, i.e., placing a relatively higher importance on past values.

Chapter 7

ENERGY LEVEL INACCURACY IMPROVEMENT

7.1. Introduction

In this chapter, we will look at methods to improve energy overall inaccuracy levels. Based on the knowledge age inaccuracy level analysis in Section 5.5.4, we propose techniques to deal with old data. We propose three techniques: Guessing (Section 7.2), Prediction (Section 7.3) and Smart Prediction (Section 7.4).

All proposed methods are compared to Default OLSR. We refer to OLSR with QoS-related state propagated, and using the default parameters (Hello-interval 2, TC-interval 5, MPR-coverage 1 and TC-redundancy 0) as "Default OLSR".

7.2. Overall Inaccuracy Level Using Guessing

As shown so far, the way OLSR functions imposes a high level of inaccuracy in the network. Increasing the overhead of the OLSR protocol to get more accurate view of the nodes in the network seems to work only under low traffic rates. This leads us to very important questions before analyzing how to improve the accuracy levels resulting from the characteristics of the OLSR protocol. Does OLSR have a potential for QoS routing? And is it worthwhile propagating the energy level information?

We propose a *Guessing* mechanism in which instead of propagating the energy level information, each node will guess the energy level of the other nodes based on its own energy consumption. We assume the networks under consideration are homogeneous, in which all the nodes are configured with the same wireless card and therefore use the same energy model (same transmitting, receiving and idle power consumption) and they start with the same initial energy level, so every node assumes that the energy level of the other nodes in the network is the same as its own energy level.

7.3. Overall Inaccuracy Level Using Prediction

Based on the energy level knowledge age inaccuracy level analysis described in Section 5.5.5, the results have shown that the main source of inaccuracies is the existence of really old data. The idea here is "What if every node locally adjusts nodes' old energy levels based on their behavior pattern".

In this *Prediction* mechanism, each node locally extrapolates an expected energy level based on old energy levels and energy consumption rate pattern for all other nodes.

Every second of the simulation, instead of having every node report its perceived knowledge for every other node in the network as is, a node's perceived value is first adjusted based on its past behavior and then the adjusted value is reported.

The pseudo-code for the Prediction mechanism algorithm is given below. There are some details that need to be noticed here:

- Every node keeps a record of the last perceived values of all the other nodes and the timestamps of when this data is created.

- Whenever a node receives a new perceived value for another node, along with the timestamp, it calculates, if possible, its consumption rate. It then adjusts the perceived value based on the consumption rate.

    - Whenever a node receives a new perceived value, there are 3 cases:

        1. The new perceived value is the same as the old perceived value which means that no update was received for that node

        2. The timestamp associated with the new perceived value is smaller than the timestamp of the old perceived value. This might seem wrong at first but there

are 2 cases where that might happen. Let us say, for example, that node 1 receives knowledge about node 2 from two different sources (say "a" and "b") and that the one from "a" was the one with the most recent information
- It might be that the next time node 1 receives a message from "a", the message does not include any info about 2 (2 is no longer heard), so the tuple expires from the information base and we are left with the older information (the only available knowledge corresponding to a tuple for node "b")
- The second case is that the messages were received out of order, so the message with the later information was received first. If within 15 seconds no further info is received, the tuple expires and again the tuple with the older info is used until it expires (expiration time is calculated from the time the knowledge is received)

3. The timestamp associated with the new perceived value is higher than the timestamp of the old perceived value which is the typical case, since we are collecting the most recent information.

- Consumption rate refers to the node's energy consumption per second and is calculated as: (old perceived value – new perceived value) / (new timestamp – old timestamp)

- The adjusted value is calculated as the perceived value minus the amount of energy this node consumed since it sent out this value until collection time.

  Adjusted value = (perceived value – consumption rate * (collection time – perceived value timestamp))

- Since for the calculation of the overall inaccuracy level we ignore the first 50 seconds and start data collection at time 50, at time 50 no adjustments are made and the adjusted values are the same as perceived values since consumption rates are not known yet.

- We only use perceived measurements that are at least 1 second apart to update the consumption rates since we are interested in long-term rates, not potentially extreme rates that could result from a node continuously sending packets for a short period of time and it happened to be between successive protocol messages, for instance.

```
NUMBEROFSIMULATIONS := 10;
NUMBEROFNODES := 50;
STARTINGTIME := 50;

while (! end of simulation) do
begin
  for (each pair of nodes (n1, n2)) do
  begin
    if (time == STARTINGTIME)
       report adjusted values same as perceived values
    else
       if (n2 old perceived value timestamp == n2 new perceived value timestamp)
          if a consumption rate is known for n2
             adjust the new perceived value based on the consumption rate
          else
             just report the perceived value since consumption rate is not known yet

       else if (n2 new perceived value timestamp < n2 old perceived value timestamp)
          if (n2 new perceived value timestamp  == 0) //the node went down
             report n2 new perceived value and;
             reset the consumption rate to 0
          else //Just older information
             if (the absolute difference between n2 new and old perceived value timestamps >= 1 )
                compute the consumption rate;
                adjust n2 new or old perceived value based on the calculated consumption rate and;
                report the adjusted value
             else
                if a consumption rate is known for n2
                   adjust n2 old perceived value based on the consumption rate and;
                   report the adjusted value
                else
                   report n2 old perceived value

       else //n2 new perceived value > n2 old perceived value
          if (n2 old perceived value == 0) //the node was down
             report n2 new perceived value
          else
             if (the absolute difference between n2 new and old perceived value timestamps >= 1 )
                compute the consumption rate;
                adjust n2 new perceived value based on the calculated consumption rate and;
                report the adjusted value
             else
                if a consumption rate is known for n2
                   adjust n2 new perceived value based on the consumption rate and;
                   report the adjusted value
                else
                   report n2 new perceived value
  end
end
```

Since the number of times an adjustment takes place is an indicator of how well our prediction algorithm works, Table 24 below shows the percentage of times an energy level adjustment takes place under the different traffic rates.

Traffic Interval	Percentage of Number of Adjustments
0.2	97%
0.14	95%
0.09	91%
0.04	82%
0.02	78%

Table 24: Percentage of number of times adjustment takes place under different traffic rates

It can be easily noticed from this table that the adjustments percentage is much lower under higher traffic rates than it is under lower traffic rates. This is due to two main factors. First, for any adjustment to take place we need two different perceived value readings so that a consumption rate can be calculated and used to adjust the perceived value. But under high traffic rates and with the high levels of message loss and delay, updates are not received in a timely fashion. And the second factor is that under high traffic rates nodes get disconnected more frequently and consequently they get reset. When a node becomes heard again, it is dealt with as a new node and again two different perceived value readings are needed for the adjustments to start taking place.

7.4. Overall Inaccuracy Level Using Smart Prediction

The only apparent drawback of the Prediction algorithm is the need to wait for two different perceived values readings, so a consumption rate can be calculated and used to adjust the perceived values.

In this section we propose the *Smart Prediction* algorithm which is an enhanced version of the prediction algorithm so that adjustments take place almost all the time.

In the Smart Prediction algorithm, for every pair of nodes (n1, n2), if n2's consumption rate is not known yet, n1 adjusts the perceived value of n2 based on the average of all known consumption rates for other nodes. If n1 knows not a single consumption rate for other nodes, it adjusts n2's perceived energy level based on its (n1) consumption rate.

Using all known nodes' consumption rates eliminates the domination of outliers and ensures closeness to the actual consumption rate, assuming again that nodes are somewhat homogeneous in their wireless cards. Unlike the Guessing approach, we do not assume or require that all nodes have the same initial energy level.

7.5. Default OLSR vs. Guessing vs. Prediction vs. Smart Prediction

In this section, we first show the overall inaccuracy level resulting from the three different strategies described in Sections 7.2, 7.3 and 7.4. We then compare Prediction, Smart Prediction and Default OLSR (OLSR using the default parameters with the QoS-related state propagated). We also compare the Guessing technique to Default OLSR to evaluate whether the OLSR protocol is a good candidate for QoS routing.

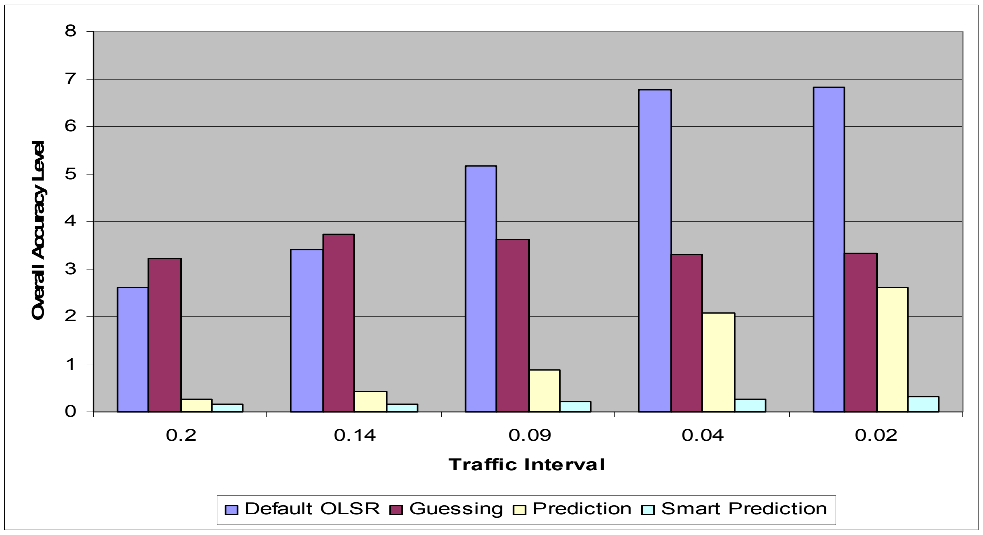


Figure 26: Overall Inaccuracy Level using Default OLSR vs. Guessing vs. Prediction vs. Smart Prediction

According to Figure 26, our Prediction algorithm improves the overall inaccuracy level under different traffic rates. The improvement under higher traffic rates is not as high as it is under lower traffic rates. As discussed in Section 7.3, for an adjustment to take place, a node must wait for two different perceived values. But under high traffic rates, due to message loss and delays, the percentage of times adjustments take place is much lower.

Since the Smart Prediction algorithm takes care of the problem of not being able to adjust the energy level perceived value all the time, it achieves much better performance in terms of overall inaccuracy level, especially under higher traffic rates.

Figure 26 clearly shows that both the Prediction and the Smart Prediction algorithms outperform the Default OLSR protocol. At the same time, the Smart Prediction algorithm outperforms the Prediction algorithm in improving the overall inaccuracy level.

Although according to Figure 26, at first glance it seems that propagating the energy level information is better than Guessing under low traffic rate, whereas under higher traffic rates, Guessing outperforms the default OLSR. However, the fact that under the Guessing technique a node assumes that the energy level of all the other nodes is the same as its own energy level leads to increasing the gap between the guessed and the actual value over time.

Figures 27 – 31 looks at the overall inaccuracy level as a function of time under traffic intervals of 0.2, 0.14, 0.09, 0.04 and 0.02 (respectively). It is clear that the overall inaccuracy level drifts linearly with time for Guessing. And as traffic rate increases, it needs more time for the Guessing technique to cross the Default OLSR. This means that the results in Figure 26 are misleading and if we run the algorithm a little longer under the high traffic rates, Default OLSR will outperform Guessing. As result, even with the unrealistic assumption of nodes having the same initial energy level under the Guessing technique, Default OLSR outperforms Guessing under all traffic rates.

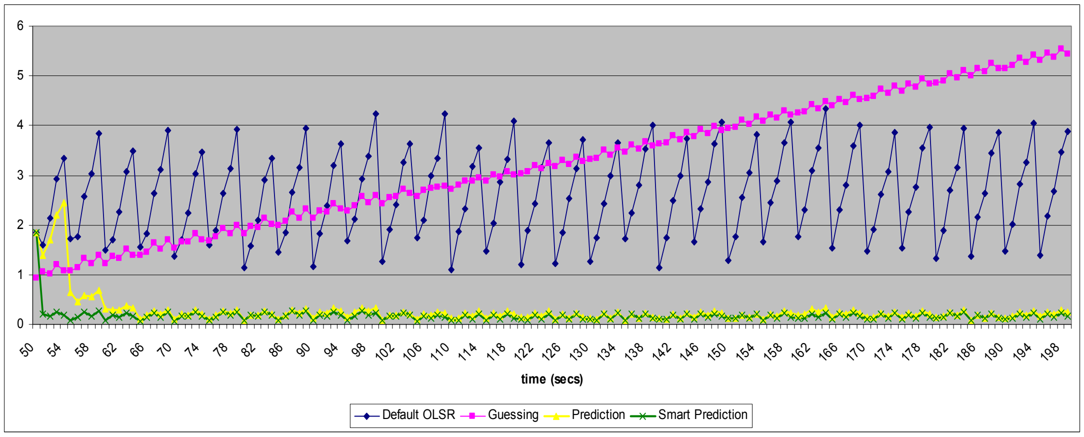


Figure 27: Overall Inaccuracy Level as a Function of Time under 0.2 Traffic Interval

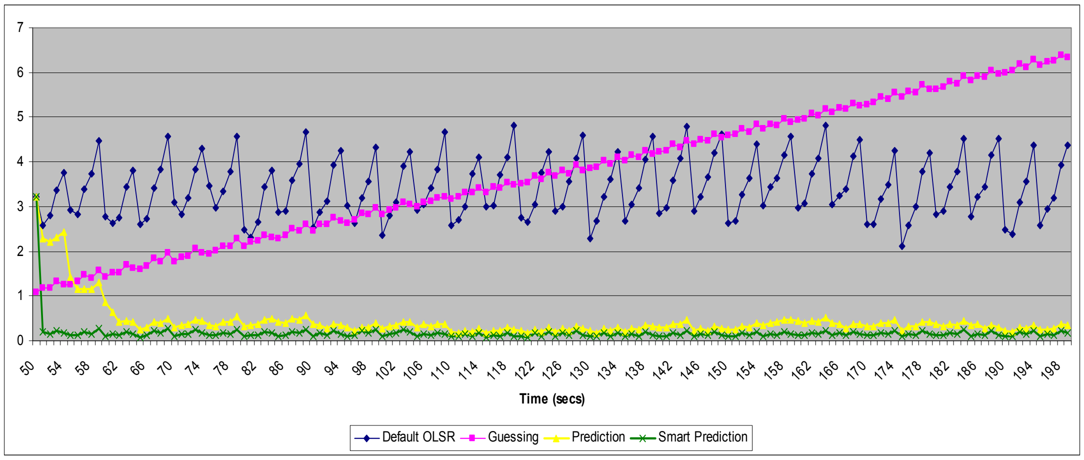


Figure 28: Overall Inaccuracy Level as a Function of Time under 0.14 Traffic Interval

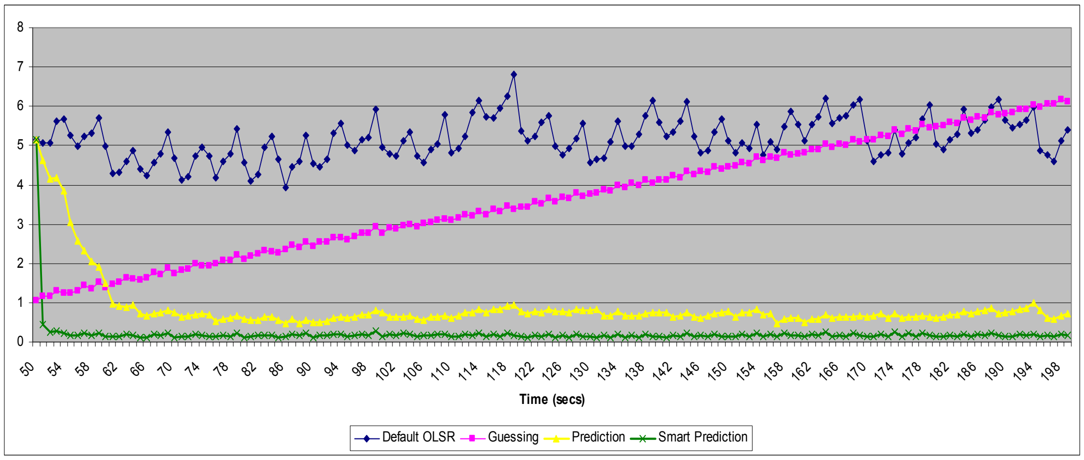


Figure 29: Overall Inaccuracy Level as a Function of Time under 0.09 Traffic Interval

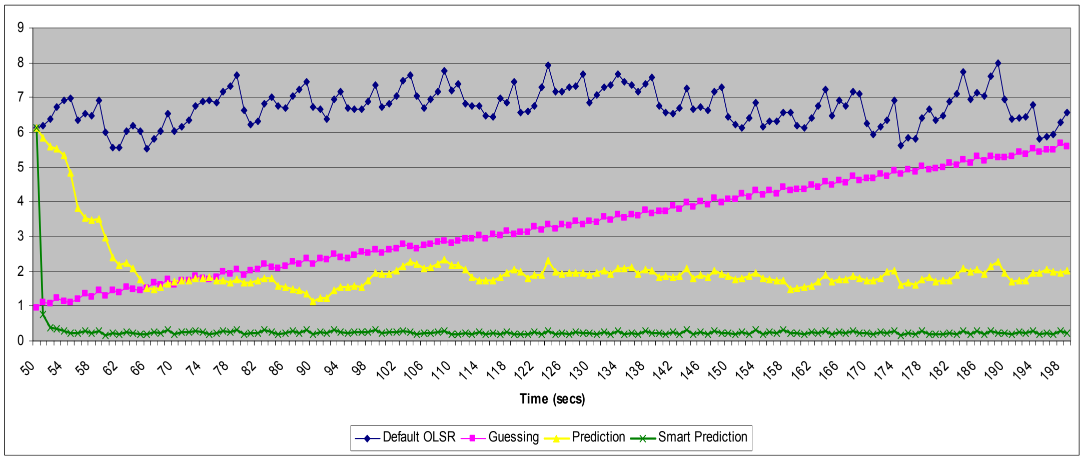


Figure 30: Overall Inaccuracy Level as a Function of Time under 0.04 Traffic Interval

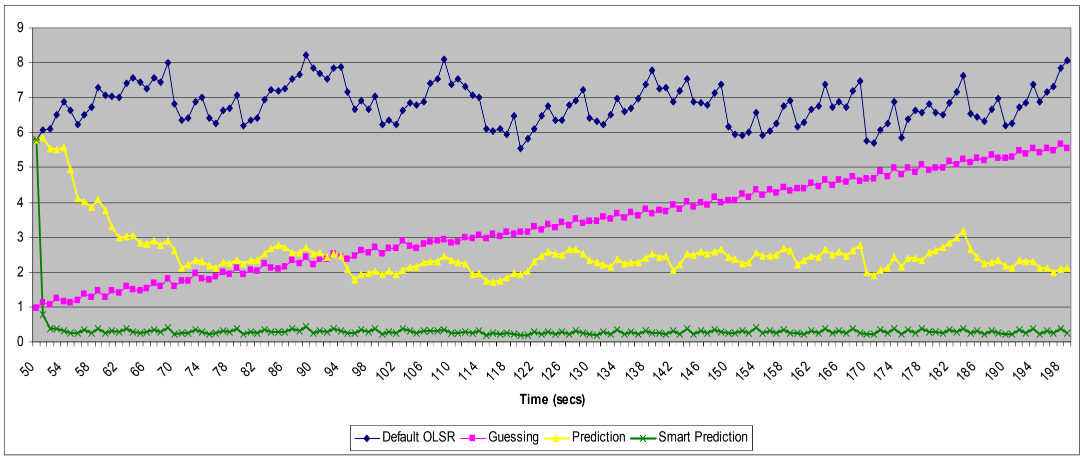


Figure 31: Overall Inaccuracy Level as a Function of Time under 0.02 Traffic Interval

It is important to point out that both the Prediction and Smart Prediction algorithms have the same overhead since the same messages are sent. So we only look at overall inaccuracy levels to compare performance. On the other hand, the Guessing technique does not require all the additional fields in the Hello and TC messages (the energy levels and timestamps). Therefore, it induces less Byte overhead since smaller messages are sent. However, it is the number of message rather that the size of the messages that greatly impact the overhead of the network.

7.6. Conclusion

One of the main concerns when collecting QoS information, is whether it is worthwhile propagating this information with all the associated overhead. We proposed a *Guessing* mechanism in which instead of propagating the energy level information, each node guesses the energy level of the other nodes based on its own energy consumption. The results show that under all traffic rates, default OLSR eventually outperforms Guessing. Overall inaccuracy level for Guessing tends to drifts linearly with time. If a node is very active, it consumes more energy and hence it overestimates the consumption rates of other nodes and vice versa, if a node is less active, it consumes less energy and consequently, it underestimates the consumption rate of other nodes. As a result, the difference between the guessed and the actual value increases over time.

Based on the characteristics of the energy level as a monotonically decreasing parameter and the knowledge age analysis, two techniques were proposed to reduce energy level inaccuracies, Prediction and Smart Prediction. Under the Prediction technique, a node's energy level is adjusted based on its past behavior (its own consumption rate). But due to message loss and delays, a node's consumption rates cannot be calculated all the time, since a node needs two different perceived value readings to calculate the consumption rate of another node. And hence, energy level adjustments do not take place all the time.

Smart Prediction is a modified version of the Prediction technique such that, if no behavioral pattern was known for a node then its energy levels is adjusted based on the average of all known consumption rates for other nodes. This reduces the chances of the domination of a single consumption rate.

The results show that both Prediction and Smart Prediction outperform the Default OLSR solution. Moreover, Smart Prediction outperforms Prediction since energy levels are adjusted all the time.

Chapter 8

CONCLUSION AND FUTURE WORK

In this work, we used the Optimized Link State Routing (OLSR) protocol as the underlying MANET routing protocol. We reported the quantification of state information accuracy for two metrics, queue length and energy level. We defined state information accuracy as the average difference between the perceived QoS-related state and its actual state.

For each of our metrics, we determined the inaccuracy level under different traffic rates. It was shown that traffic induces a high overall inaccuracy level to the network. And as traffic rate increases, the overall inaccuracy level increases. As the traffic rate increases, the network becomes more congested. Since network buffers have limited capacity in terms of storage and processing of arriving packets, this affects significantly the performance of the network, causing long delays and high loss rate. Consequently, information available to the nodes in the network becomes outdated and no longer accurate.

Moreover, as nodes send/receive packets at a higher rate, they experience a higher rate of queue length change (in case of queue length), or they consume energy at a higher rate (in case of energy level), contributing to the higher overall inaccuracy levels.

Tuning the protocol parameters did not seem to have a noticeable impact on overall inaccuracy level for both the queue length and energy level metrics. Therefore, other approaches were considered to improve inaccuracy levels.

Number of hops analysis concluded that very little improvement in energy level inaccuracy can be obtained under low traffic rates by using a combination of Hello-interval 1 and TC-interval 3 for OLSR parameters. In addition, knowledge age analysis for both queue length and energy level concluded that inaccuracy levels are highly affected by the age of the data. It suggested that any proposed solution should deal with outdated knowledge.

Accordingly, for the queue length, two techniques were proposed to reduce inaccuracies: A Probing technique in which whenever a node has old knowledge about another node, it sends that node a probe message requesting a state information update; and an Exponential Averaging technique in which a parameter is used to smooth short-term fluctuations and to highlight long-term trends. Under the Probing technique, although probe messages did provide a high percentage of more recent state information, it was not sufficient to improve inaccuracy level since the information provided was still relatively old under different threshold values of knowledge age. The increased levels of overhead resulting from the extra messages sent caused longer delays and higher loss rate, contributing to this unexpected behavior of our probing technique. Similarly, the Exponential Averaging technique did not positively impact the overall inaccuracy level. On the contrary, the trend was towards less accurate queue length information as older data was weighted more.

For the energy level, a Guessing technique in which a node guesses the energy level of other nodes based on its consumption rate showed that propagating the energy level information, using Default OLSR, provided the nodes in the network with more accurate energy level information. Overall inaccuracy level using Guessing tends to drift linearly with time, increasing the gap between the guessed and actual energy level values.

Two other techniques were proposed to reduce energy level inaccuracies, Prediction and Smart Prediction. Under the Prediction technique, a node's energy level is adjusted based on its past behavior (its own consumption rate). On the other hand, Smart Prediction is a modified version of the Prediction technique such that, if no behavioral pattern was known for a node, its energy level is adjusted based on the average of all known consumption rates for other nodes. The results show that both Prediction and Smart Prediction outperform the Default OLSR (OLSR with QoS-related state propagated, and using default parameters). Moreover, Smart Prediction outperforms Prediction since energy levels adjustments take place all the time. In addition to that, the overheads associated with Prediction and Smart Prediction is exactly the same as the Default OLSR since no extra messages or fields are required.

This work can be pursued further in some areas:

- One of the apparent problems with the Probing technique is that when a probe message is retransmitted, it gets relayed by a single specified node. Hence, there is a high probability that the message will be lost because of a broken link or a buffer overflow (at that node). The Probing technique could be enhanced to ensure that a probe message will be propagated more reliably. Hopefully that will improve the queue length inaccuracies.

- Also, at this point, with no success in improving the queue length inaccuracy level. Can we "prove" that this is the best accuracy level that can be obtained for such a rapidly fluctuating parameter under the associated traffic rates?

- The only restriction under the Smart Prediction technique is that nodes are assumed to be homogeneous in their wireless cards. As part of future work, Smart Prediction could be evaluated when nodes have heterogeneous wireless interfaces. In other words, nodes are not required to have similar transmitting, receiving or idle power.

- Overall inaccuracy levels can be determined under different mobility patterns to get a broader understanding of the different factors that might affect the state information accuracy.

- Even though other metrics, such as bandwidth and delay, cannot be measured exactly, the associated inaccuracies could be investigated to see how they will behave. Will they follow the systematic inaccuracy observed for the energy level, or random inaccuracy like the queue length metric?

- Based on our implementation, QoS-related state information is now included in the protocol massages only. What if we piggyback this information on all messages, including both control and data messages. Will that provide more accurate information since we will not be limited to learning from control messages only?

- As the first step towards supporting QoS routing, our work reports the quantification of state information accuracy and proposes new techniques to reduce inaccuracies. The proposed techniques for queue length did not have an impact on queue length overall inaccuracy level, whereas the techniques proposed for the energy level did improve the energy overall inaccuracy level.

  But how effective is providing this more accurate information on the routing decision? Will providing more accurate energy level information help making better routing decisions compared to the Default OLSR solution? For example, will Prediction and Smart Prediction calculate the routes that are closest to the optimal routes, the routes with the highest remaining energy, compared to the Default OLSR? Similarly, even though the probing technique did not seem to positively impact the queue length inaccuracy level, does that mean that probing will cause poorer routing decisions compared to the Default OLSR?

  According to the work done in [9], having more accurate QoS-related state information, via more frequent state updates, does improve the performance of the QoS routing. Therefore, we strongly believe that inaccuracy has an impact on QoS routing performance, but how accurate does the information have to be. Thus, a more in-depth study on the impact of inaccuracy levels on the QoS routing performance could be conducted.

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat, S. Tripathi, *Quality of Service Based Routing: A Performance Perspective*, ACM SIGCOMM, Vancouver, BC, Canada, Volume 28, Issue 4, Pages 17-28, October 1998.

[2] B. Awerbuch, D. Holmer, H. Rubens, *High Throughput Route Selection in Multi-Rate Ad Hoc Wireless Networks*, Wireless On-demand Network Systems (WONS 2004), Pages 251-268, 2004.

[3] J. H Chang, L. Tassiulas, *Energy Conserving Routing in Wireless Ad-hoc Networks*, IEEE INFOCOM, Tel Aviv, Israel, Volume 1, Pages 22-31, March 2000.

[4] T. Clausen, P. Jacquet, *Optimized Link State Routing Protocol*, IETF Mobile Ad Hoc Networks Working Group, IETF RFC 3626, Oct. 2003.

[5] T. Clausen, P. Jacquet, L. Viennot, *Investigating the Impact of Partial Topology in Proactive MANET Routing Protocols*, The 5th International Symposium on Wireless Personal Multimedia Communications, Volume 3, Pages 1374-1378, October 2002.

[6] S. R. Das, C. E. Perkins, E. M. Royer, *Performance comparison of two on-demand routing protocols for ad hoc networks*, INFOCOM 2000. 19th Annual Joint Conference of the IEEE Computer and Communications Societies. Volume 1, Pages 3-12, March 2000.

[7] D. S. J. De Couto, D. Aguayo, J. Bicket, R. Morris, *A High-Throughput Path Metric for Multi-Hop Wireless Routing*, 9th ACM International Conference on Mobile Computing and Networking (MobiCom '03), San Diego, California, Pages 134-146, September 2003.

[8] L. M. Feeney, M. Nilsson, *Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment*, IEEE INFOCOM, Anchorage AK, Volume 3, Pages 1548-1557, April 2001.

[9] Y. Ge, T. Kunz, L. Lamont, *Proactive QoS Routing in Ad-Hoc Networks*, The 2nd International Conference on Ad-Hoc Networks and Wireless, Montreal, Canada, October 2003.

[10] R. A. Guerin, A. Orda, *QoS routing in networks with inaccurate information: theory and algorithms*, IEEE/ACM Transactions on Networking, Volume 7, Issue 3, Pages 350-364, June 1999.

[11] Qi Han, N. Venkatasubramanian, *Information collection services for QoS-Aware mobile applications*, IEEE Transactions on Mobile Computing,, Volume 5, Issue 5, Pages 518-535, May 2006.

[12] Hipercom Project, *OOLSR Implementation of the OLSR Optimized Link State Routing Protocol*, http://hipercom.inria.fr/oolsr/, November 2004.

[13] D. B. Johnson, D. A. Maltz, Y-C Hu, *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, IETF Mobile Ad Hoc Networks Working Group, IETF RFC 4728, February 2007.

[14] S. J. Lee, M. Gerla, *Dynamic load-aware routing in ad hoc networks*, IEEE International Conference on Communications, Volume 10, Pages 3206-3210, June 2001.

[15] C. E. Perkins, P. Bhagwat, *Highly Dynamic Destination Sequenced Distance Vector Routing for Mobile Computers*, ACM SIGCOMM '94 Conference on Communication Architectures, Protocols and Applications, London, UK, Pages 234-244, 1994.

[16] C. E. Perkins, E. M. Royer, S. R. Das, *Ad Hoc On-demand Distance Vector (AODV) Routing*, IETF Mobile Ad Hoc Networks Working Group, IETF RFC3561, July 2003.

[17] T. Plesse, J. Lecomte, C. Adjih, M. Badel, et al, *OLSR Performance Measurement in a Military Mobile Ad-hoc Network*, 24th International Conference on Distributed Computing Systems Workshops, Pages 704-709, October 2004.

[18] S. H. Shah, K. Chen, and K. Nahrstedt, *Dynamic Bandwidth Management for Single-Hop Ad Hoc Wireless Networks*, ACM Mobile Networks and Applications, Volume 10, Issue 1-2, Pages 199-217, February 2005.

[19] A. Shaikh, J. Rexford, K. G. Shin, *Evaluating the impact of stale link state on quality-of-service routing*, IEEE/ACM Transactions on Networking,, Volume 9, Issue 2, Pages 162-176, April 2001.

[20] S. Singh, M. Woo, C. S. Raghavendra, *Power-Aware Routing in Mobile Ad Hoc Networks*, ACM Mobile Computing and Networking Conference, Dallas, Texas, Pages 181-190, October 1998.

[21] C.-K. Toh, *Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks*, IEEE Communications Magazine, Pages 2-11, June 2001.

[22] A. Tonnesen, *Implementing and Extending the Optimized Link State Routing Protocol*, Master Thesis, Department of Informatics, University of Oslo, August 2004.

[23] P. E. Villanueva-Pena, T. Kunz, and P. Dhakal, *Extending network knowledge: Making OLSR a Quality of Service conductive protocol*, International Conference on Communications and Mobile Computing, Vancouver, Canada, Pages 103-108, July 2006.

[24] K. Wu, J. Harms, *Load-Sensitive Routing for Mobile Ad Hoc Networks*, 26th Annual IEEE Conference on Local Computer Networks, Pages 568-575, 2001.

[25] Zhou, H. Hassanein, *Load-Balanced Wireless Ad Hoc Routing*, Canadian Conference on Electrical and Computer Engineering, Volume 2, Pages 1157-1161, May 2001.