AN ABSTRACT OF THE THESIS OF

Hassan Hussein Sinky for the degree of Master of Science in Computer Science presented on February 26, 2010.

Title: Implementation and Performance Measurement and Analysis of OLSR Protocol

Abstract approved: _____

Bechir Hamdaoui

This paper provides a measurement-based performance evaluation of the Optimized Link State Routing (OLSR) protocol. Two versions of OLSR, OLSR-ETX and OLSR-ETT, are implemented and evaluated on a mesh network that we built from off-the-shelf commercial components. OLSR-ETX uses the Expected Transmission Count (ETX) metric whereas, OLSR-ETT uses the Expected Transmission Time (ETT) metric as a means of assessing link quality. The paper describes our implementation process of the ETT metric using the plug-in feature of OLSRd, and our calculation method of link bandwidth using the packet-pair technique. A series of measurements are conducted in our testbed to analyze and compare the performance of ETX and ETT metrics. Our measurements show that OLSR-ETT outperforms OLSR-ETX significantly in terms of packet loss, end-to-end delay, and stability, yielding a much more robust, reliable, and efficient routing.

Implementation and Performance Measurement and Analysis of OLSR Protocol

by

Hassan Hussein Sinky

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented February 26, 2010
Commencement June 2010

Master of Science thesis of Hassan Hussein Sinky presented on February 26, 2010.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electric Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Hassan Hussein Sinky, Author

ACKNOWLEDGEMENTS

First and foremost I would like to thank God for giving me the strength in fulfilling my goals and ultimately making what I achieved possible. I would like to give thanks to all those who have supported me in any respect during the completion of my degree. In particular my parents, my brother and sister for their continued support. In my daily work I have been blessed with a great group of fellow students who provided affective feedback and constructive suggestions throughout the course of my research. Lastly this thesis would not have been completed without the guidance, encouragement and support of my advisor, Dr. Bechir Hamdaoui.

TABLE OF CONTENTS

LIST OF FIGURES

DEDICATION

For my family, who offered me unconditional love and support throughout the course of this thesis. To my father, Hussein Sinky, who taught me that nothing can be accomplished without hard work and dedication. To my mother, Faykah Juma, who taught me to love what I am doing in order to be successful. To my brother and sister, Mohammed Sinky and Tassnym Sinky, who always brought joy and happiness during difficult times.

Chapter 1 – INTRODUCTION

The convenient features of wireless technology, such as mobility, portability, and ease of use and deployment, are the main reasons behind the technology's tremendous success. Wireless mesh networks (WMNs) have specifically been designed to take advantage of these wireless features [5]. WMNs are known for their self-configuration ability to form a network on power-up, for their easy installation and maintenance, and for their cost-effectiveness. Mesh nodes may act as sources/clients when they themselves generate data traffic, or as relays/routers when they forward traffic for other nodes through multi-hop routing. When a route breaks due to a node's (or a link's) failure, nodes can auto-recover by rediscovering an alternate routing path without the intervention of a central unit or an administrator. WMNs are also cost effective as they eliminate the need for a core network. That is, nodes no longer require a wireless router to connect to each other since each node acts as the client and as a router, thus reducing the number of components that need to be purchased as well as the network setup costs.

In this paper, we implement, measure and evaluate the performance of the Optimized Link State Routing (OLSR) protocol [4] on a mesh network that we recently built from off-the-shelf commercial components. OLSR is a pro-active, table-driven, link-state routing protocol for mobile and wireless multi-hop networks, and as defined in RFC 3626 [4], it uses hop-count as the metric for computing shortest paths. In this work, two versions of OLSR are implemented and evaluated: OLSR-ETX and OLSR-ETT. OLSR-

ETX uses the Expected Transmission Count (ETX) metric whereas, OLSR-ETT uses the Expected Transmission Time (ETT) metric as a means of assessing/determining link quality. Our measurements show that OLSR-ETT outperforms OLSR-ETX significantly in terms of packet loss, end-to-end delay, and stability, yielding a much more robust, reliable, and efficient routing.

The rest of the paper is organized as follows. We begin by giving a brief literature review in Chapter 2. We describe our wireless mesh network testbed in Chapter 3. We then, in Chapter 4, present OLSR and its routing metrics. In Chapter 5 we introduce the ETT routing metric, and cover its implementation process with OLSR. Using our deployed testbed, in Chapter 6, we perform a series of tests to evaluate and compare the performance of our implementation of the ETT metric (OLSR-ETT) with that of the ETX metric (OLSR-ETX). In Chapter 7, we present and analyze our results. Finally, we conclude the paper in Chapter 8.

Chapter 2 – Literature Review

Wireless mesh networks consist of mobile devices or nodes connected by wirless links. Due to the random nature of these networks reliable and efficient routing protocols must be implemented. The topology of such networks (physical connectivity) may randomly change due to many factors such as node mobility, resource constraints and link quality. Protocols used in wired networks cannot be applied to wireless mesh networks due to their dynamic toplogy, decentralized configuration and bandwidth and resource constrained nodes [6]. Thus, optimized and efficient routing protocols are an essential aspect in wireless networks responsible for finding paths followed by data packets from a source node to a destination node. A variety of routing protocols specific to wireless mesh networks have been proposed such as Dynamic Source Routing (DSR), Ad Hoc On Demand Distance Vector Routing (AODV) and Optimized Link State Routing (OLSR).

There are three categories of routing protocols; proactive, reactive and hybrid. Proactive protocols share information with the nodes in a network on a regular basis or over a specific time interval. The advantage to this method is the minimal delay when acquiring routes since they have already been calculated and routing tables populated. The disadvantage to this method, however, is that it requires more power consumption due to constant route calculations. Reactive protocols, on the other hand, share topology information with other nodes in the network on an on-demand basis or when a route request

is made. The advantage to this method is that it requires less power consumption due to less computations. However, the disadvantage is that an initial delay is incurred when acquiring a route since calculations must be made and routing tables populated before hand. Finally, hybrid protocols combine the best features of both proactive and reactive. With this protocol, nodes that are within a certain distance of the node concerned are said to be in the routing zone. A proactive approach is used with nodes within the routing zone whereas a reactive approach is used with nodes outside the routing zone [6].

There are two general responsibilities carried out by these routing protocols; to form the topology of a network by detecting mobile devices and to calculate the best possible path from a source node to a destination node based on the quality of the path. Our research focused on the Optimized Link State Routing protocol (OLSR). Mainly improving link quality sensing used by the routing protocol. Link quality, also referred to as link cost or link weight, can be determined by a combination of factors such as reliability and throughput. Once the quality of a link is assessed and a weight or cost is associated with it the best possible route can be calculated. Using the information acquired from the routing protocols, a routing algorithm is used to actually perform the calculations. The weight or costs associated with the links in a network are used by the routing algorithm to decide which route is shorter (better). A node then uses the topology information gathered by the routing protocols and routing algorithms to compute next hop destinations using shortest paths. Routing tables are then populated using the results.

There are many types of routing algorithms today capable of computing the shortest path. Among these routing algorithms is the popular Dijsktra's algorithm conceived by

Dutch computer scientist Edsger Dijkstra in 1959. This graph search algorithm solves the shortest path problem based on non-negative link weights or costs. Given a source vertex the algorithm computes the shortest path from that vertex to all other vertices in the graph. The algorithm can also be used to find the shortest path from a source vertex to a single destination vertex where it stops when the shortest path to the destination vertex is found. For example, let's assume that the nodes within a graph represent cities and the weights of the edges between a pair of nodes represent the distances between the cities. Given a starting city, Dijkstra's algorithm could be used to calculate the shortest route from that city to every other city in the graph. Thus, Dijkstra's algorithm is particulary suitable and widely used by routing protocols to solve the shortest path problem. The following is pseudocode for Dijkstra's algorithm [9]:

Algorithm 1 DIJKSTRA$(Graph, source)$

```
1:  N ← source
2:  for all nodes v in Graph do
3:      if v adjacent to source then
4:          Dist(v) ← cost(source, v)
5:      else
6:          Dist(v) ← infinity
7:      end if
8:  end for
9:  loop
10:     find w not in N such that Dist(w) is a minimum
11:     add w to N
12:     for all v adjacent to w and not in N do
13:         Dist(v) ← min(Dist(v), Dist(w) + cost(w, v))
14:     end for
15:     until all nodes are in N
16: end loop
```

In our research, Optimized Link State Routing (OLSR) was used as the routing protocol. OLSR is a proactive, table-driven and Dijkstra based protocol. It is particularly suitable for large and dense networks. That is, the number of nodes within such networks can be increased and OLSR would still perform efficiently. The protocol will be discussed in detail later on. The following chapter discusses our wireless mesh network testbed.

Chapter 3 – An Experimental Network: The Testbed

We designed and built a wireless mesh network on the third floor of the EECS building at OSU. Each node (i.e., wireless router) consists of an Alix.3C2 board [3] with a 500MHz AMD Geode processor and 256MB DDR RAM. The board uses a 512MB compact flash card for internal storage, and has two mini-PCI slots, two USB ports, and an Ethernet jack. Each board is also equipped with a Wistron NeWeb CM9 radio card for wireless connectivity. It is based on the Atheros AR5004 chipset so it is compatible with the driver software used, and easy to setup. A large number of wireless modes are also supported, allowing a wide variety of test scenarios and connectivity options. It supports IEEE 802.11a/b/g, IEEE 802.11g Super Mode, and IEEE 802.11a Turbo Mode. It is also highly configurable with WPA and WEP security options, transmission power control, and dynamic frequency selection support. This card provides enough features for the current implementation as well as for future improvements, expansions, or tests.

Voyage Linux, a distribution based off of Debian Linux, is the operating system (OS) of choice. Because of its Debian heritage, software installation and configuration is easy to handle. It requires 128MB of hard drive space and is best suited for network appliances such as firewalls, wireless access points, routers and network storage devices. This Linux distribution is also designed specifically to run on the Alix boards and similar hardware.

MadWifi [1] wireless drivers provided with the Voyage Linux distribution were used

for communication between the wireless card and the OS. They already support Atheros based wireless cards so there were no implementation conflicts.

In addition, installed on each wireless device is a software based implementation of the Optimized Link State Routing (OLSR) protocol (OLSRd, v. 0.5.5) [2]. OLSRd's sole responsibility is to detect neighbors, determine the quality of a link and populate the routing tables of the wireless devices. The network and OLSRd are configured on the nodes to run using a startup script. Depending on the chosen configuration, OLSRd can be set to run in one of two modes: standard OLSR (i.e., default hop-count metric) (OLSR-HOPS) or OLSR with ETX (OLSR-ETX). In our experiments and evaluations presented in Chapters 6 and 7 we focus mainly on OLSR-ETX. Each node is configured with a web server, `lighttpd` for analyzing the values associated with route calculation and node detection. `lighttpd` is chosen as the host web service because of its basic functionality and light system requirements. The following section briefly covers node detection and route calculation performed by OLSRd as well as the metrics used to determine the quality of a link.

Chapter 4 – Optimized Link State Routing

OLSR (Optimized Link State Routing) protocol is a proactive, Djikstra's algorithm-based routing protocol for mobile, ad-hoc mesh networks [4]. OLSR also provides an optional extension to include and account for link quality information in determining shortest paths. It uses ETX (Expected Transmission Count) to assess a link's quality, where ETX is the average number of transmissions/retransmissions required to successfully send a packet from a source to a destination.

In this section we cover OLSR's node detection and route calculation mechanisms as well as why the ETX metric may not be enough to assess link quality.

4.1 Node Detection

Each node detects its direct and two-hop neighbors by broadcasting hello messages. Once a list of neighbors is obtained a subset of nodes from this list are selected as multi-point relays or MPRs. MPRs are solely responsible for forwarding information regarding their MPR selectors throughout the network. This is what is called selective flooding since only a subset of nodes is forwarding messages, thus, greatly reducing the amount of messages in the network. The nodes which are selected as an MPR by some neighbor nodes announce this information in their control messages. Thereby, a node announces to the network that it has reachability to the nodes which have selected it as an

MPR. Topology control (TC) messages are used to share this information with all other nodes in the network. TC messages are sent periodically; i.e., they might not be sent if there are no updates and sent earlier if there are updates. Each node maintains topology information about the network acquired from TC messages and is used for routing table calculations. In essence, these messages contain nodes' IDs, their neighbors, and link qualities.

4.2 Route Calculation

OLSR-HOPS calculates routes purely based on least number of hops; i.e., it does not account for link reliability, nor link throughput. Thus, the assumption made by OLSR-HOPS is that all link throughputs are identical across the entire network. Consider applying OLSR-HOPS for determining the routes from node A to node E in the network example given in Fig. 4.1. Since OLSR-HOPS selects the routes with the least hop count, Route 2 (A $\rightarrow$ C $\rightarrow$ E) is always selected in lieu of Route 1 (A $\rightarrow$ B $\rightarrow$ D $\rightarrow$ E).

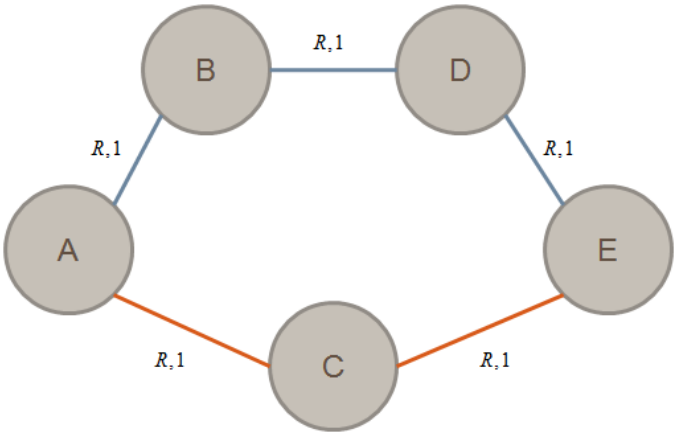

Figure 4.1: Shortest path routing metric: OLSR-HOPS

The problem with OLSR-HOPS's least hop count metric is that a path may still be chosen even when it has higher packet loss and/or lesser end-to-end throughput than other paths. In fact, this metric assumes that there is no packet loss, and that throughputs are identical across each link. However, in wireless networks, packet losses are inevitable, and hence, not accounting for these losses can lead to poor routing performance. To overcome this, OLSRd has been extended to use the Expected Transmission Count (ETX) quality metric as a means for accounting for packet loss.

The method used by OLSRd to obtain the ETX value is through the use of hello messages, link qualities (LQ) and neighbor link qualities (NLQ). As each node sends out hello messages to find and detect their direct neighbors, LQs and NLQs can be calculated based on the fraction of packet loss (while hello messaging), thus the probabilities of a successful transmission. So, LQ assesses how good a given link is in the direction from a node's neighbor to the node itself, and NLQ assesses how good a given link is in the direction from the node to the node's neighbor. These values can be communicated back and forth between the node and its neighbor through hello messages. Once these direct neighbors are found, the ETX value for a node and its neighbor is calculated using the link's quality (LQ) and the neighbor's version of the link quality (NLQ); by accounting for LQ and NLQ, the ETX value is the same for both directions, and is

$$ETX = \frac{1}{LQ * NLQ} \tag{4.1}$$

TC messages are then used to share this information among all nodes in the network. The cost of a certain path/route is then simply the summation of the ETX values along

the path/route. Thus, the route with the smallest ETX sum is chosen, representing the least number of transmissions it takes to get a packet from the source to the destination.

Let's consider the same network example as shown in Fig. 4.2. Assume p_1 and p_2 are the probabilities that a packet is successfully transmitted over a link belonging to Route 1 and Route 2, respectively. The corresponding ETX values are then simply $1/p_1$ and $1/p_2$. OLSR-ETX finds the paths with the shortest paths but while accounting for packet retransmission, or alternatively, finds the paths with the least end-to-end delay[1]. A packet of size L bits experiences a delay of $\frac{3L}{p_1 R}$ and $\frac{2L}{p_2 R}$ when respectively traveling Route 1 and Route 2, where R bits/sec is the data rate supported on each link. Thus, OLSR-ETX chooses Route 2 over Route 1 if and only if $p_2 > \frac{2}{3} * p_1$. By ignoring throughput, OLSR-ETX assumes all links have identical throughputs across the network. Thus, the L and R variables are negated in the resulting route condition previously mentioned.

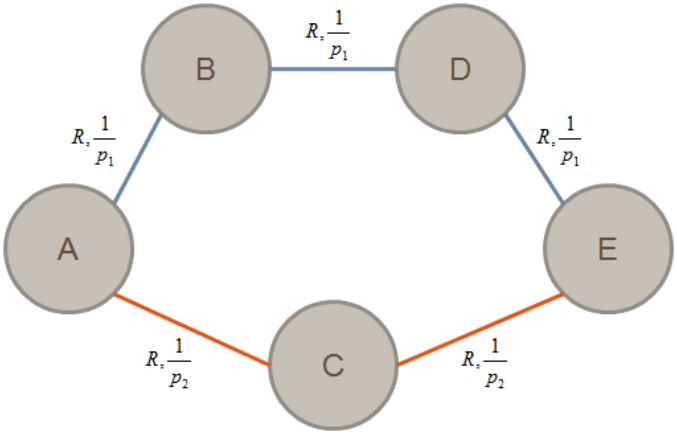

Figure 4.2: ETX routing metric: OLSR-ETX

[1]Hereafter, we consider transmission delays only; we ignore all other types of delays.

We can see that the path chosen by OLSR-ETX now depends not only on the hop count, but also on ETX, and hence, the least number of hops may not always be better.

4.3 Why is ETX not Enough?

Recall that ETX represents the average number of transmissions/retransmissions needed to successfully deliver a packet over a link. Hence, by incorporating ETX as its link-quality metric, OLSRd takes into account the links' reliability when deciding which paths to choose. By accounting for the links' reliability, ETX tends then to find robust routes. ETX, however, does not take into account links' data rates. When links' data rates are not accounted for, a short path with lower ETX may be chosen over another longer path with higher ETX albeit the latter may be able to support a higher overall throughput and less end-to-end delay. We, therefore, introduce a new link-quality metric, called the Expected Transmission Time (ETT), calculated as the ratio of ETX to the link's data rate; this new metric represents the inverse of the expected data rate of the link giving us the expected time a packet takes to successfully be sent. With this new metric, a less reliable link can then be part of the shortest path if it supports high enough data rates. ETT is then more suitable for, and effective in, networks with heterogenous transmission data rates, such as cognitive radio networks [8, 11, 13].

By ignoring throughput, the ETX metric assumes that the links along a path have identical throughputs. However, different paths and links may support different data rates. Consider the same example as before only this time the throughput across links on each route is different, and represented by R_1 and R_2 as illustrated in Figure 4.3. The

end-to-end delay on Route 1 and Route 2 is now $\frac{3L}{p_1 R_1}$ and $\frac{2L}{p_2 R_2}$, respectively, and OLSR-ETT chooses Route 2 over Route 1 if and only if $p_2 R_2 > \frac{2}{3} * p_1 R_1$. We can see that the path chosen now depends on, and accounts for, three factors: reliability, throughput, and hop count.

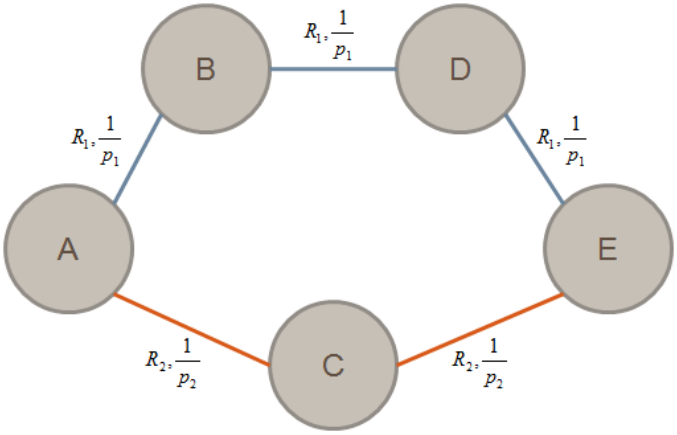

Figure 4.3: ETT routing metric: OLSR-ETT

The following section defines the ETT metric, and covers the implementation process of incorporating it with the OLSR routing protocol installed on the wireless nodes.

Chapter 5 – ETT Calculation and Implementation

As stated previously, currently OLSRd uses the ETX metric to compute and determine the quality of a link by monitoring the expected number of transmissions it takes to successfully send a packet. However, we want to improve this by calculating the bandwidth of a link and implementing it with ETX giving us the Expected Transmission Time (ETT) routing metric:

$$ETT = ETX * \frac{L}{R} \tag{5.1}$$

We can see from the equation above that ETT considers three factors in its calculation; ETX, throughput and, since the weight of a path is equated to the summation of the ETT values across the path, the hops are also considered. In short, the ETT value represents the expected time it takes to successfully transmit a packet from the source to a destination. By implementing ETT within the OLSR daemon, wireless mesh networks will improve in reliability, stability and efficiency. To tackle this task OLSRd's plug-in feature was used to create a plug-in responsible for computing a link's bandwidth and merging it with OLSRd's ETX. The process of computing the bandwidth of a link involves a technique known as the packet-pair technique.

5.1 Packet-Pair Technique

The packet-pair technique uses two packets of the same size sent back to back from the source to a destination. Both packets are traveling from the same sending node to the same receiving node. Synchronization problems arise when the system times on the wireless devices, required to compute the bandwidth, are not the same. The inter-arrival time of the two packets on the receiving node can be used to accurately compute the bandwidth of the link even when the sending and receiving nodes are not synchronized, thus masking the synchronization effect.

Let t_0 and t_1 be the arrival times of the first and second packets respectively at the destination and s be the size of the second packet. The link bandwidth b can then be calculated using the following equation [10]:

$$b = \frac{s}{t_1 - t_0} \tag{5.2}$$

In this work, we implemented the packet-pair technique above into the default routing mechanism of OLSRd. This technique is incorporated into OLSRd to calculate the throughput supported by each link, which is then used to compute the ETT metric as given in Equation (5.1).

5.2 Implementation

One of the advantages of OLSRd is that it provides a convenient plug-in interface for users. A plugin can be created to access OLSRd data structures without modifying OL-

SRd's main code. So the first design choice was to implement our metric modifications using an ETT plug-in. The sole responsibility of the plug-in would then be to compute the bandwidth using the packet-pair technique and to communicate the results back to the source so that an average bandwidth could be calculated. An exponential weighted moving average is then maintained for each neighbor node. These values are shared amongst neighbors as well as other nodes within the network.

The default forwarding or broadcasting mechanism used by OLSRd may not be sufficient enough for calculating a link's throughput. Computing the bandwidth with simply broadcasting the packet-pair probes and forwarding messages is inaccurate since the broadcast uses the IEEE 802.11 basic physical rates [12]. To resolve this issue, the use of inter-process communications (IPC) is required. That is, creating a separate socket, aside from the main socket used by OLSRd, for the plug-in to communicate with the other nodes. This required the creation of a new thread apart from the main OLSR thread where the plug-in will have a life of its own. By using IPC we bypass OLSRd's forwarding/ broadcasting mechanism and focus only on node to node communication for a more accurate bandwidth calculation.

To reduce the complexity we focused on the immediate one-hop/ symmetric neighbors of a node. Each node would then need to conduct the packet pair technique only with each of its direct neighbors to obtain the bandwidth for their links. The bandwidth can then be easily incorporated with the ETX metric and shared with other nodes in the network using OLSR's current implemented mechanism. The following section explains the testbed configuration and experiments used to evaluate the performance of our ETT OLSRd modifications (i.e, OLSR-ETT) as well as OLSRd (i.e, OLSR-ETX).

Chapter 6 – Topology Configuration and Testing

In order to evaluate OLSR-ETX and OLSR-ETT, a number of tests are performed to characterize the network performance. The 802.11g network is first tested for functionality. All the routing tables of the nodes are observed and shown to be populating, proving that the wireless network and OLSRd are properly functioning, and all the nodes are communicating with each other.

Tests were conducted over a two week span. Each test on average took anywhere between 10 minutes to 8 hours to complete, depending on the test variables. Wireless interferences in the EECS building are naturally fluctuating; some days we would see little interference, and on busy work days we would experience much more interference. A 54 Mbps theoretical link would often only achieve 5-10 Mbps in practice due to the interference and multiple networks present in the building. Network channels are auto-set, switching from channel to channel, meaning there was no way to set the testbed on an un-interfered channel. Due to the amount of interference present in the building, the number of nodes was reduced to seven and brought closer together to avoid and minimize inaccurate calculations and interference related issues. The topology was configured in a way such that there were only two paths available from node A (source) to node G (destination). The data rates for the shorter path were compromised and set to 1 Mbps while data rates for the longer path were set to 54 Mbps as illustrated in Figure 6.1. All of the tests were performed from node A through the terminal interface

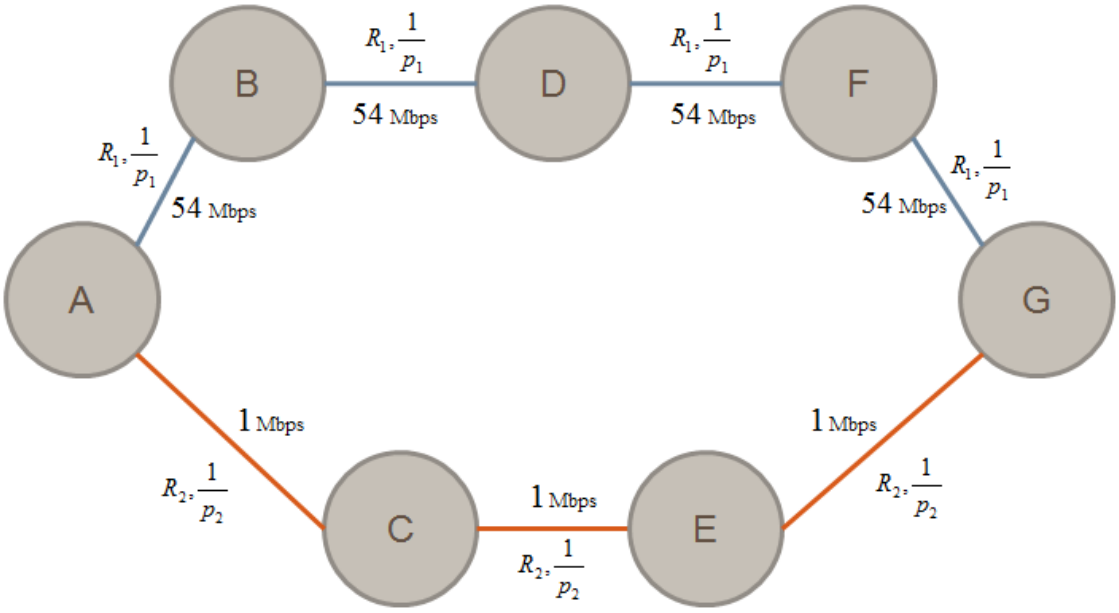

Figure 6.1: Test Topology Configuration

to node G. Laptops were used for both nodes A and G for analyzing all data regarding our testbed such as routes selected, packet loss, round-trip times and briefly streaming video. Both laptops have OLSRd running and are part of the mesh network.

Upon the completetion of the OLSR-ETT plugin a preliminary test was conducted to quickly visualize the benefits of OLSR-ETT over OLSR-ETX. A video file was streamed from node A to node G. This was done once while using the OSLR-ETT protocol and once with the OLSR-ETX protocol. With OLSR-ETT the video was received at the destination as a smooth watchable stream. On the other hand, using the OLSR-ETX rendered the stream unwatchable. This was encouraging since the implemented OLSR-ETT was correctly detecting faster links as opposed to OSLR-ETX and thus providing a higher quality stream.

In addition, a series of ping tests were conducted as a more concrete method to measure network performance. Ping is a widely available network administration utility used to detect if a host is reachable on a network. Results of the ping are summarized and displayed once complete such as packet loss and round-trip times. Not only can Ping be used for testing host reachability it can also be used for recording the route taken by a ping and flooding the network with requests and replies. Ping operates by sending an ECHO-REQUEST packet, a ping, to the destination host and waiting for an ECHO-REPLY, known as a pong. If a response is not received or has timed-out the ping packet is considered as lost. The round-trip time of a ping is timed and recorded when an ECHO-REPLY is received otherwise a lost ping is not calculated into the average round-trip time. When flooding the network, ping packets are output as fast as they return or 100 times per second, whichever is more.

Our first performance test involved *varying the number* of pings flooded/ injected into the network and measuring the percentage of packet loss as well as average round-trip time while fixing the size of the pings to 300 bytes. The second test involved fixing the number of pings flooded into the network to 100,000 and *varying the size* of the pings from 100 to 1000 bytes. The following section illustrates the results measured by our tests on the network in Figure 6.1.

Chapter 7 – Performance Measurement and Analysis

Prior to performing the tests our assumptions were that OLSR-ETT would provide much shorter round-trip times and less packet loss compared to OLSR-ETX due to its ability to find faster paths and handle larger amounts of data. With OLSR-ETX we experienced instability in our network in the form of frequently changing routes and high packet loss rates. In our configuration, OLSR-ETX would consistently choose the shorter path regardless of the poor link speed. Due to the rate of flooding causing large amounts of network congestion on the poor links the shorter routes would occasionally be lost and the alternate, faster route would be chosen. But as soon as the shorter path returned, OLSR-ETX would revert back to it as its route of choice. The frequent route changes and drops influenced network instability causing performance to fluctuate and hence degrade. With OLSR-ETT routes were much more stable and hardly ever changed resulting in less packet loss and shorter round-trip times. These tests measure the performance of the network, as a result of the metrics used (OLSRD-ETX or OLSR-ETT), when under a lot of stress.

Our first set of tests yielded impressive results. Figure 7.1 illustrates that as the *number* of pings flooded into the network increases the percentage of packet loss with OLSR-ETX increases at a faster rate than OLSR-ETT with each incremental test. OLSR-ETX packet loss ranged from 2 to 51 percent. However OLSR-ETT packet loss only ranged from 0 to 2 percent. Similarly we can see from Figure 7.2 that as the number of

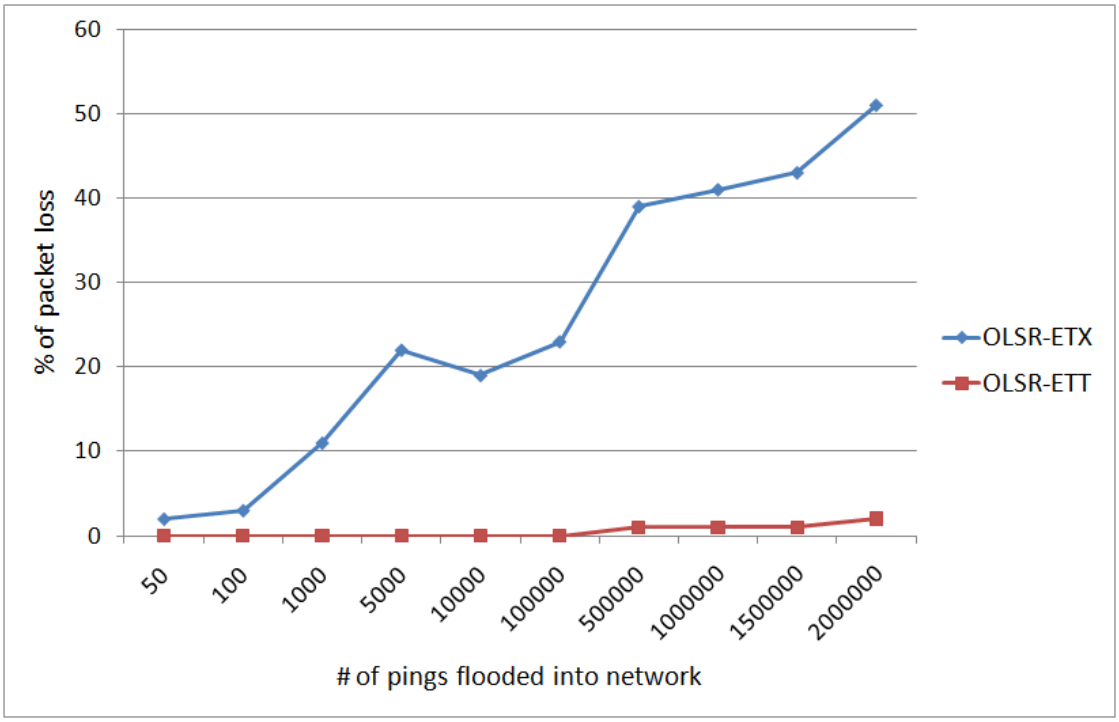

Figure 7.1: Packet loss as a function of the number of 300-byte pings

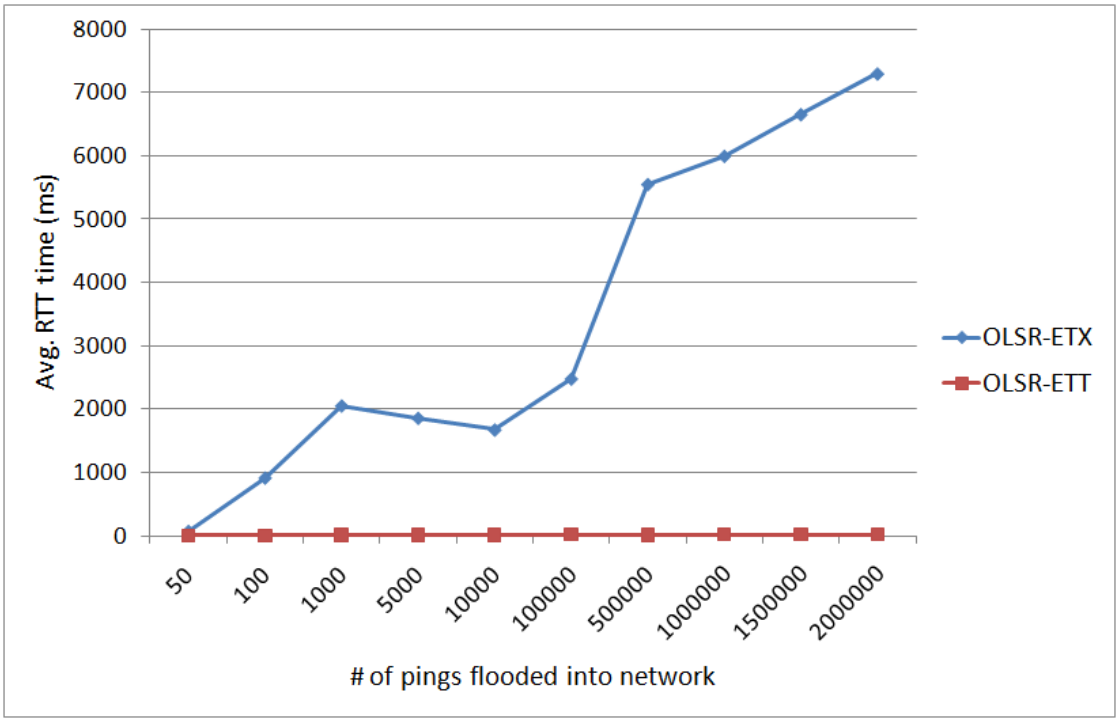

Figure 7.2: Average RTT as a function of the number of 300-byte pings

pings flooded into the network increases the average round-trip time of successful pings increases rapidly with OLSR-ETX whereas OLSR-ETT increases at a much slower rate. OLSR-ETX round-trip times varied from 78.1 to 7300.4 milliseconds. OLSR-ETT round-trip times only ranged from 3.67 to 18.76 milliseconds.

Our second set of tests yielded the following results. Varying the *size* of the pings flooded using OLSR-ETT we observed a much more stable and efficient network. Packet loss rarely exceeded 1 percent as shown in Figure 7.3. OLSR-ETX would experience high packet loss while varying the size of the pings. However, aside from the 100 byte test, packet-loss remained consistent compared to the previous tests ranging from 23 to 43 percent. As a result of OLSR-ETT we were left with a network that saw little to no route changes and much less round-trip times. Round-trip times remained somewhat consistent throughout the testing process, ranging from 6 to 14 milliseconds, as shown in Fig. 7.4. This was not the case with OSLR-ETX where round-trip times ranged from 132.08 to 2785.3 milliseconds.

As we can see from our tests OSLR-ETX's performance with our testbed resulted in very erratic behavior. Both scenarios can be explained as OLSR-ETX's lack of ability to detect faster routes. With a much slower path chosen the amount of pings flooding that specific route will overwhelm the devices as they try to keep up with the demand. Packets fill up the queues at a faster rate causing the devices to drop packets and increase delays. Essentially, regarding the slower path, packets are entering faster than they are leaving. OLSR-ETX's inability to detect and avoid bottleneck routes greatly affects the performance of the network. The degradation in performance was at its peak when using OLSR-ETX where tests took anywhere from minutes to hours to complete. This delay

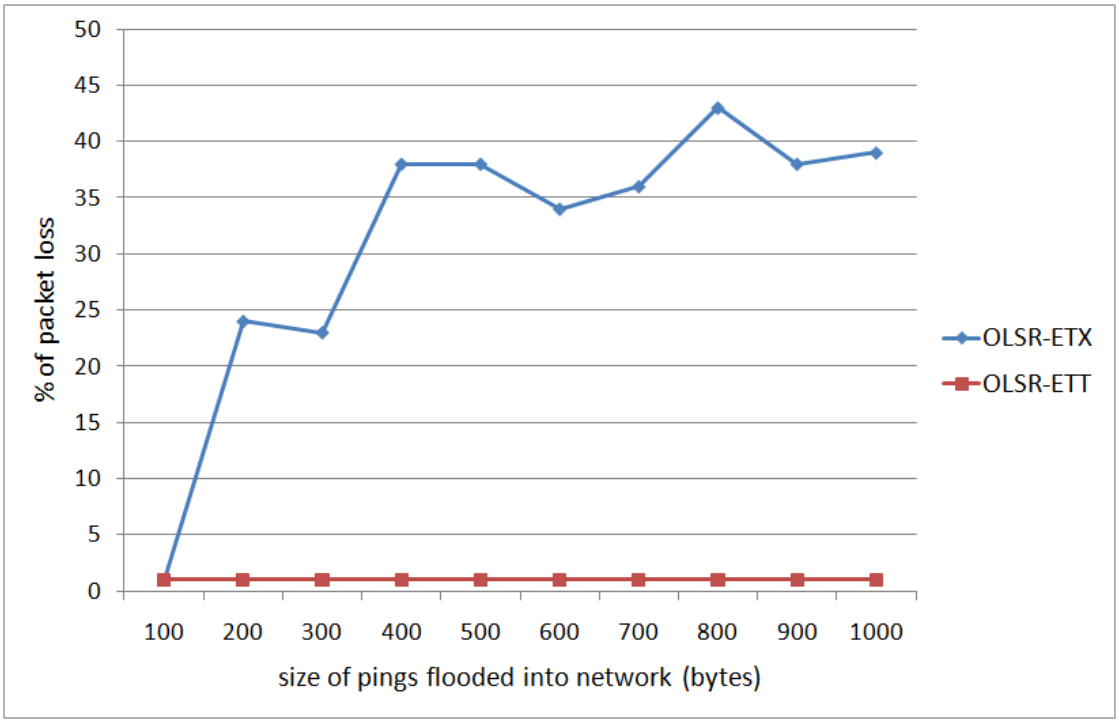

Figure 7.3: Packet loss measured as a function of ping sizes

was present in OSLR-ETT however not nearly as much. Although this is a drastic test configuration, our results show the importance of, in addition to accounting for packet loss and shortest paths, the ability to detect faster links; improving the performance of a network. With the amount of data being transmitted wirelessly, links may fluctuate in speed, thus it is a must to account for these changes.

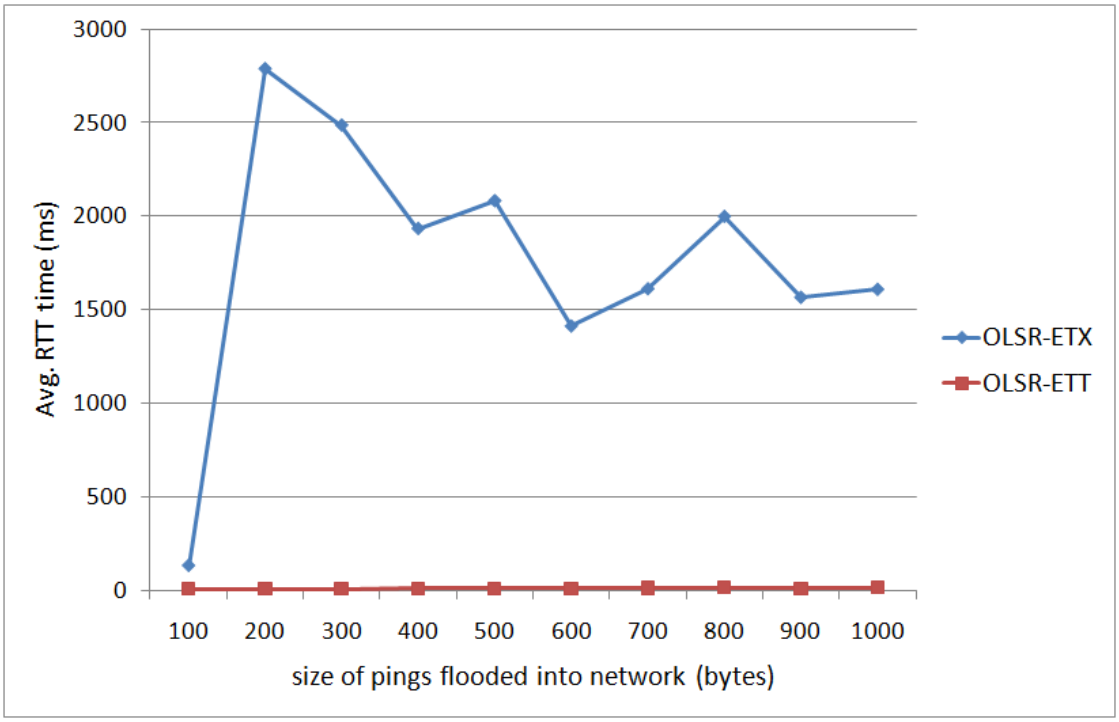

Figure 7.4: Average RTT measured as a function of ping sizes

Chapter 8 – Conclusion and Future Extension

This work implements the Optimized Link State Routing (OLSR) protocol on a wireless mesh network, recently built from off-the-shelf commercial components at Oregon State University, and evaluates its performance. Two versions of OLSR were implemented, tested, and evaluated: OLSR with ETX (OLSR-ETX) and OLSR with ETT (OLSR-ETT). In theory, the inclusion of bandwidth detection should result in a routing metric much more in tune with the natural behavior of wireless networks. Our measurements show that OLSR-ETT outperforms OLSR-ETX significantly in terms of packet loss, end-to-end delay, and stability.

As a future work, we intend to extend OLSR-ETT implementation to support multi-channel-capable networks. Since different channels are likely to support different data rates, OLSR-ETT may be well suited for wireless mesh networks that are capable of multiple channel access, enabling then more robust and efficient routing.

Bibliography

[1] *www.madwifi.com*.

[2] *www.olsr.org*.

[3] *www.pcengines.ch*.

[4] *www.ietf.org/rfc/rfc3626.txt*.

[5] R. Bruno, M. Conti, and E. Gregori. Mesh networks: commodity multihop ad hoc networks. *IEEE Comm. Mag.*, March 2005.

[6] B. S. Manoj C. Siva Ram Murthy. *Ad Hoc Wireless Networks*. Prentice Hall, 2008.

[7] Dbora C. Muchaluat-Saade Luiz C. Schara Magalhes Diego Passos, Doublas Vidal Teizeira and Clio V. N. Albuquerque. Mesh network performance measurements. 2005.

[8] B. Hamdaoui and K. G. Shin. OS-MAC: An efficient MAC protocol for spectrum-agile wireless networks. *IEEE Transactions on Mobile Computing*, July 2008.

[9] Keith W. Ross James F. Kurose. *Computer Networking*. Addison Wesley Longman, Inc., 2001.

[10] Kevin Lai. Packet pair technique. `http://www.hpl.hp.com/personal/Kevin_Lai/projects/nettimer/publications/usits2001/node2.html`, April 2001.

[11] L. Ma, X. Han, and C. C. Shen. Dynamic open spectrum sharing MAC protocol for wireless ad hoc networks. In *IEEE Int'l Symp. on New Frontiers in Dynamic Spectrum Access Networks*, 2005.

[12] Igor M. Moraes Lus Henrique M. K. Costa Otto Carlos M. B. Duarte Pedro Miguel Esposito, Miguel Elias M. Campista and Marcelo G. Rubinstein. Implementing the expected transmission time metric for olsr wireless mesh networks. July 2008.

[13] S. Wu, C. Lin, Y. Tseng, and J. Sheu. A new multi-channel MAC protocol with on demmand channel assignment for multi-hop mobile ad hoc networks. In *Proceedings of Inter. Symp. on Parallel Architectures, Algorithms and Networks*, pages 232–237, 2000.